COVOX

VOICE MASTER®
junior

USER MANUAL

COMMODORE 64
COMMODORE 128 IN 64 MODE
INCLUDES 128 MODE PLAYBACK

Software Version VMJC 1.Ø.

Includes:

SPEECH RECORDING AND PLAYBACK
SPEECH WORD RECOGNITION
TALKING CLOCK, CALCULATOR, BLACKJACK
SPEECH AMPLITUDE EDITOR
WHISTLE TO MUSIC WRITING
WHISTLE TO MUSIC PERFORMING
PRINTED PROGRAM EXAMPLES
EXAMPLES OF INPUT/OUTPUT THROUGH USER PORT
HOW TO PREPARE TAPE CASSETTE SYSTEM
MACHINE PROGRAM FOR SOFTWARE AUTHORS

Copyright 1987 by
COVOX, Inc.
675 Conger Street
Eugene, Oregon 97402

First Printing October, 1987

ADDENDUM: THE SCREEN BLANKS DURING RECORDING AND HENCE '?' IS NOT USEABLE FOR SETTING GAIN. USE EITHER THE 'DANCING BARS' DESCRIBED ON PAGE 2 OR THE CALIB COMMAND ON PAGE 5.

CONTENTS

Product Outline....1
Setup and Demonstration Programs....1
. CLOCK....3
. CALCULATOR....3
. BLACKJACK....3
Recording and Playback with BASIC....4
The BAR program....7
Concepts in Recognition....8
Recognition Programming....9
Error Criteria, Thresholds, and Hints....12
Template Making....14
The PLAY Program....16
Amplitude Editor....17
Music Program: TRACKER....19
Music Program: COMPOSER....21

APPENDICES

A. BACKUP....28
B. CASSETTE SYSTEM....29
C. AUDIO ALTERNATIVE....31
D. COMMAND SUMMARY....32
E. MEMORY DATA AND LOCATIONS....34
F. MISCELLANEOUS INFORMATION....37
. Disk Programs and Files....37
. Phonetic Alphabet....38
. Airman's Numbers....38
. Telephone Operator's Numbers....38
G. MEMORY MAP OF C-64 WITH VM....38
H. SELECTED PROGRAMMING EXAMPLES....38
. Talking Numbers....38
. Two Approaches to a Talking Keyboard....41
. The Cash Register Vocabulary....42
I. PROGRAMMING WITH EXTERNAL EVENTS....43
. Talking Alarm System....45
. Voice External Control....46
. A "Rube Goldberg" Scheme....50
K. 128 MODE PLAYBACK....51

LIMITED WARRANTY STATEMENT

COVOX, Inc. guarantees the VOICE MASTER JUNIOR to be free from defective materials and workmanship for a period of one year from the date of purchase. COVOX, Inc. will make repairs under this warranty when the defect occurs under normal use, provided the unit is returned to the factory via prepaid transportation, and provided that the defect can be demonstrated to be due to an inherent defect of a component part or in the assembly process at the factory. Information in this manual and associated software are provided on an "as is" basis. COVOX, Inc. disclaims liability for direct, indirect, or incidental damages arising from the use of this product, including but not being limited to interruption of service, loss of business or potential profits, legal actions, or other consequential damages.

Control of environmental factors by means of voice could expose the user to some risk. Word recognition remains an unreliable technology due to uncontrollable variations in the way that normal speech is produced in an uncertain and noisy acoustic environment. Covox, Inc. specifically disclaims liability as stated in the preceding paragraph when applied to word recognition.

PATENTS AND COPYRIGHTS

The software supplied with VOICE MASTER is copyrighted. It may not be copied, reproduced, translated, or reduced to any readable medium or code for other than personal use without prior written permission of COVOX, Inc.

The hardware/software system comprising the COVOX VOICE MASTER JUNIOR is subject to existing patent applications. Unauthorized duplication for commercial purposes or to otherwise avoid payment of appropriate royalties or license fees will be deemed to be a violation of proprietary rights under patent and trademark laws.

The names COVOX and VOICE MASTER and the COVOX "logo" are registered trademarks and are the property of COVOX, Inc.

RESTRICTIONS ON SOFTWARE USE

Software may generally not be used in programs which are sold or otherwise distributed in violation of copyright laws. There is one exception. Speech that has been produced with Voice Master software may be put into other programs along with playback software, without royalty charges provided (1) software is not for commercial sale, and (2) the source of the speech must be given on the disk jacket, instruction book, and in the disk program itself in sufficient detail to permit a user to acquire a Voice Master. Those wishing to use recognition software and/or edited playback software in programs for sale are advised to contact Covox, Inc. for licensing information.

Product Outline.

The Voice Master Junior (VM Jr.) accomplishes several unrelated or partly related tasks. The manual has been separated into distinct parts in order to simplify the descriptive process. Three broad categories of utility software work with the hardware to provide functions as:

1. Recording speech and other sounds, with playback and optional editing.
2. Recognizing previously "trained" words.
3. Writing or performing music from whistles or tones.

In many practical programs, (1) and (2) above are combined. A speech editor program is also available with which recorded speech can be improved or otherwise modified for interesting effects on playback. Music, (3) above, is entirely independent of (1) and (2) as well as of the speech editor.

Some ability to program in BASIC is presumed, but not at an advanced level. Most demonstration programs are written in BASIC and can be listed for study. The VM Jr. disk contains essential utility software written in machine language, but an understanding of machine language programming is not required.

Important Note: This manual is written for the Commodore 64 and also for the Commodore 128 when in the "64" mode. However, the Commodore 128 can be used in the "128" mode for playback of speech vocabularies that have been prepared in the "64" mode. The vocabulary is placed in the upper 64K memory bank. See Appendix K for details.

Setup and Demonstration Programs.

It is suggested that you go through the demonstrations on the disk before becoming involved in details. But first, make a copy of the original disk and put the original away in a safe place. It is wise to not remove the write protect tab on the original. If you are not familiar with the copying procedure, consult Appendix A where the process of copying only part of your master disk is also discussed.

Although most users of the Commodore 64 will employ a disk drive, it is possible to also work from a cassette based system. We do not offer a cassette for this purpose. But we do provide instructions in Appendix B for making a suitable cassette, provided you can do this with the aid of your own disk drive.

Does your video display have audio? Or do you want audio to go elsewhere? See Appendix C.

Plug VM Jr. into joy stick port number 2. This is the rear one on the side of the keyboard. If you are using a Commodore 128, put it into the "64" mode. Use capital letters.

Adjust the volume control on VM Jr. so that, in the absence of sounds, the level light does not flicker on and

off. Adjust to avoid more than a very occasional flicker when holding it near the mouth without speaking.

Put the VM Jr. disk into the drive and type LOAD"MENU",8 and return. Then type RUN. A main MENU is presented on which there are 8 choices. Start with DEMO. A sub-menu then appears.

Press F1 and prepare to record a word or phrase. Press F1 again and start talking. The computer waits until it perceives more than an occasional click and then it begins to record. It stops recording when it perceives a long enough period of silence to qualify as the end of a word or phrase. The screen may or may not go blank after pressing F1 for the second time and before you start talking. A click could cause blanking, but if silence immediately follows this, the computer may reset and not actually start the process. The speech buffer is about 8 seconds long and recording automatically stops when the buffer is filled, whether or not you have finished talking. The contents of the buffer are lost and the word must be re-recorded.

As you talk, you will see a small "?" in the upper right hand corner. This symbol is motionless during silent periods and jitters about as you speak. If it jitters when it should not do so, check the volume on VM Jr. It may be set too high. If it does not flicker during speech, then either the volume is very much too low, you have a hardware problem, or perhaps the device is plugged into the wrong joy stick port.

While you are in the mode with the DEMO menu, the computer will every once in a while say "Covox Voice Master-ter-ter-ter" with the last syllable decreasing in amplitude like an echo. If you do not hear this, then perhaps the audio gain on your TV set or video display, with audio, is not high enough. Or if you are too impatient to wait, then press F3 and hear what you previously recorded. If you don't like what you recorded, then simply do it over again. Experiment to get the idea.

Now try pressing F5. What you recorded comes back at different amplitudes like an echo. Press F7 and your recording is repeated at different speeds

Still on the DEMO sub menu, press F8. You immediately see a set of dancing bars as you speak, with one bar to the far right that moves from zero to a large fixed value to indicate speech on/off. Try various vowel sounds, fricatives like "ss", and whistles. You will see the bar system move about as if to indicate the frequency of what you produce at the microphone. This pattern has potential value in certain forms of speech training. A set of bars is computed from speech each 0.02 seconds. After time normalizing a word to a total number of bar scans of 12, the data are employed in part for word recognition.

Now press F2 to get a quick demonstration of word recognition. You are coached to say 4 words, Covox, Voice

Master, Computer, and Finish. This serves to "train" the computer to your voice and these particular words. Then you are asked to say one of the words into the microphone and the computer is supposed to recognize what you said. You can use numbers such as one, two, three, and four and then recognize these. But, of course, the computer program will only print to the screen the original words so that, if you say "three", the computer prints "Computer".

Now press "Q" so as to quit the DEMO sub menu and return to the main MENU. We are about to play some games. But if your interests are musical, you may want to select TRACKER or COMPOSER. In this case, first go to the section of this manual concerned with music. Or if you do not want to play some more so as to gain worthwhile experience, then simply go to the next section of this manual where we get down to some of the details.

CLOCK: You are told how to set time. Then you are prompted to record words needed for the clock, or use a pre-recorded vocabulary called CVOICE. After the clock is started, you can get the time spoken out upon pressing a key. There also is an alarm clock feature. If you do elect to make your own vocabulary, follow menu instructions. But be careful in naming your new vocabulary--you don't want to give it the same name as one that is already on the disk (i.e., CVOICE).

More will be said later on making vocabularies. Any number of different vocabularies can be chosen, provided each has a distinctive name. If you attempt to save a vocabulary with a name that already represents a vocabulary on the same disk, you will get an error message. If you must use the same name, then scratch the original vocabulary and then save the new one with the same name. (There may be an inherent "bug" in Commodore software that will sometimes give trouble if you use the disk command which replaces one file with another without changing the name. It is better to scratch and then save or PUT in the case of Voice Master speech files.) In saving a vocabulary, you must replace the original Voice Master disk with a copy or a properly formatted disk that is not write protected (i.e., there must be a notch in the jacket). You can easily swap disks if you need to load a Voice Master program.

CALCULATOR: You are prompted to either create your own vocabulary of numbers, math symbols, equals, and so on, or you can select a pre-recorded vocabulary. Then you can operate the computer the same way as a simple electronic pocket calculator. The computer speaks the keys as they are pressed and then speaks the answer after you press the equals sign. Three pre-recorded vocabularies are on the Voice Master disk, one called "ENGLISH", another called "SPANISH" and a third, "GERMAN". Choose your language! (These vocabularies have been edited.)

BLACKJACK: This is a standard form of the gambling game as played at Las Vegas (but without doubles or splits). You train the computer to your words. From then on, you need not touch the keyboard. A pre-recorded (edited) vocabulary called

"DEALER" speaks cards and other data to you. You are given a sum of money to start. You say numbers such as "one", "five" to give 15. You can erase this bet by voice (say "erase") and replace it. Then say "bet" and the dealer deals and reads visible cards. You then say "stand" or "hit me". The idea is to get as close to "21" as you can without going over. Aces can count as one or eleven. After you "stand", the dealer plays out the dealer hand and and then says either that you won or he/she won or a draw and your accumulated capital is updated. Say "cards" and see what cards have been played (for "card counting" practice). Say "cards" again and return to the game.

If you don't like the dealer's voice, you can make your own vocabulary and save it. A special BASIC program, BJVOICE, is provided on the disk for this purpose. You also can save the templates that you trained the computer to recognize from your own voice.

Please note that many of the demonstration programs are more than just simple demonstrations. They are useful and practical examples of what the enterprising author of software can accomplish, whether it be in education, business, or entertainment.

Also please be aware that some of the words have been edited using the "Speech Construction Set" which requires use of the standard Voice Master with its capability to measure voice fundamental pitch. There is an editor on the VM Jr. disk that works with amplitudes alone. This will often give speech that is nearly as good as can be created with "SCS".

Recording and Playback with BASIC.

The all-important program is called VM. It contains recording/playback and word recognition. If you exit to BASIC from the main MENU, this program will be properly installed and a NEW is automatically issued to clear out BASIC programs used with the various demonstrations.

An alternative is to turn on the computer, put the disk in the drive, and without going to the main MENU, type

```
LOAD"VM",8,1
SYS 49152
NEW
```

where NEW is required in order to set pointers in the computer (but only when loading is done in this way).

A vocabulary of up to 64 words can be created by recording one word at a time as

```
LEARN n
```

where n is in the range 0-63. Play back the word with

```
SPEAK n
```

where you will get a tone beep if the word with index n was not recorded.

You can re-LEARN a word without affecting other words in the vocabulary.

During the period when the computer is recording, a "?" appears in the upper right hand corner of the screen and it should flicker with sound inputs. Recording normally ceases after the computer senses a short period of silence. A recording can be put in a hold status (talk to someone, take a breath, etc.) with the up-arrow key (not the cursor arrow). Press this key again in order to return to the recording state. (This procedure also applies to making templates for recognition or doing recognition.)

The keyboard command CALIB shows the "?" recording indicator in the upper corner of the screen the same as for LEARN. However, actual recording does not take place. This command is useful in adjusting gain setting. An automatic "time out" returns the program to the keyboard. Or return by striking any key.

Important! The gain setting on VM Jr. must be adjusted so that no more than an occasional flicker of the "?" occurs in the absence of speech, but even softly spoken speech must result in active flickering. It will be observed that the level light on VM Jr. flickers in synchronism with the "?" and can be used as an alternate level indicator. The level light is an "all or nothing" indication of the presence of sounds and does not indiciate relative levels. An occasional flicker is permissible because the computer program will reset as if no sound occurred if the sound is judged to not remain long enough to indicate the start of a valid word.

Clear out an entire vocabulary and set memory locations for the next vocabulary with

CLEAR x

where x is in the range 16-160. If CLEAR is typed without x specified (or in any event after loading VM without issuing a CLEAR), a default value of 64 applies. The value of x specifies the page number where the vocabulary begins. Small values give more room for a vocabulary but less room for a BASIC program. For a more detailed discussion, see the section on the special PLAY program.

A recording can be terminated at any time with any key. This can be useful in a noisy environment. Stopping a recording this way (rather than letting a period of silence cause this) puts the number 251 in memory location 151. Pressing RUN/STOP also terminates an ongoing recording but no special number is put into location 151. There also is a "time out" function which aborts the recording without anything saved if sounds last too long. A time out causes the number 250 to be put into location 151. (See code number listing in the section on recognition, and also comments on GET A$.)

You can change the speed of playback with SPEED y where y is in the range of 0-10 with default value of 7. (The value for y is in memory location BASE+256.) You can change the sampling rate during recording with RATE w where w is in the range 0-10 with default value 7. (The value of w for RATE is in memory location BASE+255.) The speech recording rate and playback speed must be the same if the speech is to be natural. If you change from a default value, then a command is required in order to return to the default value (unless you re-load VM).

If you want to save a vocabulary to disk, type

```
PUT"yourname",8
```

and if you want to load a vocabulary previously saved to disk, type

```
FIND"yourname",8
```

A saved vocabulary contains the starting page number as defined by CLEAR x and the RATE used in the initial recording. After getting a vocabulary from disk, rate may be changed but not the page number.

The loudness on playback can be changed with VOLUME m where m is in the range 0-15 with 15 the loudest, and also the default value.

Important: The loudness level on recording is always set to the maximum value of 15. Changing loudness levels from moment to moment during a word is achieved with the Amplitude Editor program (described in a later section). Changing with VOLUME m prior to playing back a word will change the volume the same way for the entire word.

There also is a PAUSE q command which is equivalent to a time-wasting FOR-NEXT loop. The number q is the number of 1/4 second delay increments.

A recording may be just a little clearer, especially if done at a high RATE, if the recording program does not have to compete with the screen display. (The graphics chip can be disabled so that it does not steal clock cycles from the processor.) Type SCREEN r where r is any integer and the screen goes blank during recording. To return to the non-blanked default value, type SCREEN (no number).

We have now described all of the wedged-in BASIC commands used for recording and playback. They are:

LEARN, SPEAK, CLEAR, SPEED, RATE, PUT,
FIND, VOLUME, PAUSE, SCREEN, and CALIB.

Here is a simple BASIC program that loads in a vocabulary (used with the CALCULATOR demonstration program) and speaks out the complete vocabulary including tone beeps at the end where no words were recorded.

```
10 FIND"ENGLISH",8
20 FOR J=0 TO 18
30 PAUSE 4
40 SPEAK J
50 NEXT J
60 END
```

We could change SPEED and VOLUME during the program because these commands have numbers that can be computed like the J in SPEAK J.

None of the wedged-in commands are recognizable by Commodore BASIC unless VM currently resides in memory. If VM is to be loaded in the middle of a program, then proceed as:

```
      ......
      ......
500 IF A=0 THEN A=1:LOAD"VM",8,1
510 SYS 49152:CLR
      ......
      ......
```

The vocabulary to be used is loaded in <u>after</u> the statements above. Without the IF-THEN statement, loading will loop endlessly. Getting out of this loop results on the second time through when the IF condition is not valid.

Multiple vocabularies can be loaded into a single program. However, whatever vocabulary is in main memory is erased and replaced with the one loaded from the disk.

There is a peculiarity of Commodore BASIC that is worth mentioning. A colon may be needed in some statements involving wedged-in commands. For example:

```
IF W=3 THEN:SPEAK J
```

When running a BASIC program, you can stop the program with the RUN/STOP key at any time and change playback characteristics such as SPEED or VOLUME with keyboard commands (or equivalent POKE's with the un-wedged program PLAY to be described). Then type CONT to continue.

When playback is in progress, you can press the SPACE BAR in order to restart playback from the beginning. This can help to evaluate the beginning parts of a recording. It also serves to produce novel stuttering sounds.

Note: A summary of wedged-in BASIC commands is given in Appendix D.

<u>The BAR Program.</u>

This program was described in the section of DEMO. It is a machine language program that is separate and distinct from any others (i.e., you don't have to first load in VM). It can be made operational at almost any time without erasing BASIC or other programs that may be present. You can get BAR from

the the DEMO program. Or, load it directly from the keyboard as

```
LOAD"BAR",8,1
SYS 20480
```

The bar graph appears immediately upon pressing return. Whatever program was ongoing before loading BAR is returned upon striking any key. Familiarity with BAR is helpful for improving microphone technique and assuring that speech is not being sensed during supposedly quite periods. Recall that you can also use CALIB for a quick check on gain setting or the level light on VM Jr. Or you could even use LEARN n and then re-record the testing word if so desired.

Concepts in Recognition

If a speech word is reduced to a set of comparatively simple characteristics, and if each characteristic is transformed to a graphical variation of time, then this set of time functions forms a "template" which characterizes the word. If several different words are formed into templates, the result is a catalog which can be used in the study of some unknown word. Recognition is based on the best fit, or match, of the unknown template with one in the catalog. This requires that the unknown be compared with each template in the catalog. If no comparison gives a good match, then it is implied that the unknown word is not in the catalog. If two or more good matches are found, then a decision involves uncertainty and advising the operator of this situation might be warranted.

The forgoing applies to virtually all kinds of pattern recognition such as speech, vision, radar echos, etc. Differences arise in the nature of the characteristics used to form templates and in the error criteria used to measure the closeness of a match. It may not be necessary to complete a match with every member of the catalog if one or more cues contained in the characteristics can narrow down the choices at an early stage. A process that sequentially narrows choices is sometimes called a tree pattern search. Voice Master recognition does involve a limited form of tree search in that a poor match may be indicated before the process for a given template has been completed with the process then jumping to the next template. Another form of tree search applies when sub-vocabularies of words and sub-catalogs of templates are employed. Voice Master allows for sub-vocabularies.

Speech is a partly predictable process. Time functions that make up a template are not completely independent of one another nor are values of functions unrelated from one moment in time to the next. Knowing a parameter at one instant can narrow down the possible values that can occur at a following instant. Narrowing choices in this way is somewhat similar to a "tree search". Using a slope/value representation for a function based on current and preceding values in order to help predict subsequent values, with results then formed into a (digital) code, is called linear predictive coding (LPC for short). If a particular value predicts the immediately

following value but has no predictive ability beyond that, the process is said to be a (first order) Markov process. We won't attempt to develop these concepts further here.

The dancing pattern of the bar graph provides the basic characteristic used in Voice Master recognition (although additional cues not shown on the bar graph may be used as well). Pattern shapes are measured at (approximately) 20 millisecond intervals and each individual pattern is designated with a set of 8 numbers. Each number is in the range 0-255 and requires one byte of storage. The total number of 8-number sets depends on the length of the word. The second operation in recognition is time normalization in which all patterns are reduced to 12 8-number sets. Both templates for the catalog and templates for the unknown are similarly normalized. The total number of numbers in each template is 12*8=96 (plus four more bytes for memory location data).

Pattern matching could commence at this time by simply taking differences between corresponding numbers between the unknown and each template in the catalog. A closeness score can be computed as the sum of the differences in magnitudes (or root-mean-square magnitudes). Certain weightings might be applied to the patterns according to relative importance of various parts of the pattern. The lowest score then indicates the best estimate for the unknown. A large lowest score indicates no good match. Two or more low scores indicate uncertainty. (In order to maintain proper comparative measures, stored templates must be normalized.)

In the Voice Master recognition algorithm, a variation of the matching process called "dynamic time warping" is employed. This procedure accounts for some minor differences in the way a word is said. The cues as functions of time can be moved slightly, as if the template was a rubber sheet. A word such as "hello" will then continue to give a good match even though the last syllable may be stretched out compared to that used in making the catalog template for the word.

The Voice Master allows for up to 32 templates per catalog. These may be broken into 4 sets of 8 templates. Each 8 may in turn be broken into subgroups. A tree-like search results if the first recognition from a restricted set of words then points, or "vectors", to a second set of words, and so on. Words in each set can be made very distinctive with an error being unlikely. In this way, two very similar words can be recognized reliably, provided that they occur in different subgroups and that neither subgroup will be addressed by the incorrect word.

There are two error criteria: No match good enough, or two or more good matches giving uncertainty. Both of these error criteria can be changed in a user written program.

Recognition Programming

The program VM must be in main memory. To make a template for a catalog, type

TRAIN n

where n is the index number given to the template, in the range 0-31. Unless your interests are in plausible existence or non-existence of a particular word, you will want to have a catalog of 2 or more templates. Thus TRAIN additional words. (Note: The keyboard CALIB command remains available for gain checking without recording template data.)

Suppose you have TRAIN(ed) a few words in the range 0-7. Now you present a word to the microphone for recognition. Type

RECOG

and all 32 templates are scanned for a best fit. Scanning all templates takes time. The scan can be limited to the first 8 templates with

RECOG 1

or to the second group 8-15 with RECOG 2, and so on to RECOG 4. You can scan two template groups, the first and third, for example, with RECOG 1,3 (or in reverse order with RECOG 3,1).

Note: Template numbers that were never TRAIN(ed) are quickly passed by in the scanning process. If, for example, only templates 0-7 were TRAIN(ed), then RECOG with scanning of all 32 templates would take about the same amount of time as RECOG 1. Speed-up with partitioning is most effective when templates outside the sub-group of interest have been TRAIN(ed).

What happens when you RECOG? The index number of the best match is put into memory location 151 in page zero. If the best match was, for example, for word index number 3, then the decimal number 3 will appear on the screen with PRINT PEEK(151).

What if you get no good match? A different number appears. A table of possibilities follows, including codes for recording and playback as well as those for recognition.

Loc. 151	Situation
248	Tone beep produced. RECOG when nothing was TRAIN(ed). RECOG word longer than any template word (twice as long). Repeated TRAIN word too long.
249	Tone beep produced. SPEAK a word never LEARN(ed).
250	Time-out. Number of half-second increments in Loc. 3.
251	Any key pressed during LEARN, TRAIN, RECOG, or SPEAK. To read the pressed key, use, GET A$. Exception: Space bar during playback after SPEAK resets to start of word and does not produce an error 251.

252	Speech memory full (LEARN only).
253	Speech input buffer full. About 8 seconds for LEARN. About 2 seconds for RECOG and TRAIN.
254	Min. error. No RECOG because 2 or more words too similar.
255	Max. error. No RECOG because no word close enough to qualify.

You can erase an entire set of templates, but later recover them if no re-TRAIN(ing) has been done, with

```
BLANK
```

You can blank one particular word with

```
BLANK n
```

where n is the index number of the word (range 0-31).

You can recover a particular BLANK(ed) template with

```
UNBLANK n
```

or all BLANK(ed) templates with

```
UNBLANK
```

These various manipulations can all be handled within a program. For example, suppose you produce 16 templates, 0-15, and want the program to consider only 7-13 at first, and then consider all except 11 and 12. A sequence of program steps (with error handling statements) could be as follows:

```
100 FORJ=0TO6:BLANKJ:NEXTJ
110 BLANK 14:BLANK 15
120 RECOG 1,2
130 UNBLANK
140 A=PEEK(151):IF A>253 THAN 110
 ...
 ...
200 BLANK 11:BLANK 12
210 RECOG 1,2
220 UNBLANK
230 A=PEEK(151)
240 IF A=250 THEN 400
 ...
 ...
```

In this example, note that the first recognition is repeated if either a MIN or MAX error occurs. The second recognition jumps elsewhere if a time-out occurs.

The nature of the number in location 151 can be most useful. A simple comparison might be: Is or is not the word (or other sound) in the catalog? Then you don't care what is in location 151 unless it is the number 254 or 255. If you pressed a key to put code 251 into location 151, then with a

GET A$ you find out which key was pressed and in this way you have created a means to mix voice and keyboard commands in the same program. With judicious handling of RECOG and various error and indicating numbers in location 151, forms of artificial intelligence can be demonstrated.

You will no doubt want to make and save a template set, and later recover it from disk memory. The commands for saving to disk and loading from disk are, respectively

```
TPUT"filename",8
TFIND"filename",8
```

The template set always contains 32 templates, even though many may never have been trained.

If a template does not provide satisfactory recognition, then it can be re-TRAIN(ed). But first it must be BLANK(ed). If you do not first BLANK, then the result will be the average of two TRAIN(ings). This is not desireable if one attempt to TRAIN was poor. But it is good practice to TRAIN each good word twice so as to average out some random errors. Averaging over three or more TRAIN(ings) may not improve recognition and can have a negative effect by muting some of the more important characteristics of a word template. But for some words, especially those without fast changing parts, multiple TRAIN(ings) can help. (Note: In re-training a word, be sure to use BLANK with an index number else you may BLANK all words in the template set.)

If you TRAIN a word more than once, and you hear a tone beep, then you have entered a word that differs in duration by 50% or more. This indicates something is abnormal. When this occurs, the number 248 is placed in location 151 as an error condition and the word just entered is not averaged. (If you are writing an original program, you might want to prompt the user to re-TRAIN.) If you re-TRAIN and continue to get beeps, perhaps your original word is at fault and you should start over again.

Error Criteria, Thresholds, and Hints.

Two kinds of errors preventing recognition were previously discussed. One error results when two or more words are too similar (254 in Loc. 151). The second is when no word in the template set is close enough to give a reasonably convincing match (255 in Loc. 151). In the process of dynamic time warp template matching, differences are accumulated between the unknown template resulting from RECOG and each and every TRAIN(ed) template being scanned. The result is a set of numbers, equal to the size of the stored template set, with values ranging from a minimum (closest match) to a maximum (poorest match). If the overall minimum score is not small, then no good match has been found. This not-small score becomes a maximum number criterion. On the other hand, if two templates show nearly the same minimum scores, then one cannot with confidence state which is the best match (allowing for noise and other uncertainties).

For most practical applications, a single number error criterion can combine both kinds of errors. The criterion is established with a wedged-in command

ACCEPT n

where n is in the range 0-4 with 0 the most lax and 4 the tightest. The default value is 2. Values for n and associated minimum and maximum numerical differences are:

n	Min	Max	
0	1	250	
1	5	200	
2	10	150	(default)
3	15	100	
4	20	100	

When a recognition is made, the best matched template word, and the accumulated error score, are to be found in memory locations in the lower part of the memory used by the Voice Master machine language code. The second best match and score are also placed in memory. These locations are:

49155	Index number closest match.
49156,49157	Error score for above (low-high bytes).
49158	Index number of second closest match.
49159,49160	Error score for above (low-high bytes).

The error criteria themselves are stored in memory as:

49161,49162	Minimum value (low-high bytes).
49163,49164	Maximum value (low-high bytes).

By PEEK(ing) the second best word and score, you can determine which word is confusing the recognition algorithm. That way, you can determine whether you need to pick another word that won't be confused, or define a new sub group.

The experienced programmer may wish to separately specify minimum and maximum error values rather then the values established by the single ACCEPT n command. Simple POKE(s) can serve to make changes.

Two other parameters are worth describing here. The general recording procedure establishes the minimum word length that will be presumed to represent a valid word. This is determined by the number of contiguous amplitude samples with non-zero values. The ruling number is in location 49165 (nominal value 10). Another parameter determines how long after the end of a word the computer must wait in order to decide when the word has in fact ended. This involves a count of contiguous amplitudes having zero values. The parameter is in location 49166 (nominal value 12). Additional data on memory locations is given in Appendix E.

Template Making.

A discussion on techniques for making good templates and achieving good recognition scores is warranted. The machine is not as good a recognizer as is an attentive human listener. (But it is vastly better than an inattentive person!) Don't ask the machine to choose between close alternatives if you would not expect a human to do well at this task. Think of how often you must ask to have numbers repeated over the telephone. Can you expect a machine to do better? The machine does have advantages, however. That is, it never tires, and it is always attentive.

Unlike a human, the machine operates with a strictly limited set of rules. One of these determines the starting and ending points of a word. Unlike the human, a simple machine does not slide a sound back and forth in time to align it with a comparison--at least not much sliding is allowed. Thus in making templates or recognizing, try to have your words start with certainty from a low noise background and end with equal certainty. Be aware of extraneous noises just before and just after speaking, such as lip smacks, tongue and teech clicks, and breath noises. You would't want recognition to depend on the way a person's false teeth click! These sounds are acoustically similar to plosive bursts and/or fricatives and could be mistaken for such. Noises at the start and/or end of a word can be especially troublesome, in part because they misrepresent just when the word is supposed to start and end. (A plosive is a brief burst of sound as in the letters "t", "k", and "p".)

VM Jr. hard-limits volume. (The standard Voice Master encodes amplitudes averaged over a period of about 0.02 second into 16 levels including zero. However, the recognition algorithm is based on hard limiting the same as with VM Jr.) Deleting amplitude cues makes the process less versatile than a human ear. Attempt to make your words at constant level and speak loud enough to get well above any background noise. If nasals are too weak by comparison, then perhaps speak with the microphone closer to the nose.

Attempt to always say your words the same way and in a natural manner. If natural, it is less likely that there will be large differences from one word to the next.

TRAIN words in the same manner and in the same environment as you expect to experience during actual RECOG. It is natural for a person to change the way speech is produced to fit the environmental situation.

A template has some random perturbations superimposed on it. Making a template that is the average of several such templates tends to smooth out these perturbations. But averaging too many tends to blur the distintive features, especially fast changing ones. The final average template will be compared to a single template from RECOG which is not averaged or smoothed. If no fast changing cue is left in the average TRAIN(ed) template, then such cues will not help in

recognition. Thus limit the number of repeated TRAIN(s) to perhaps 2. (Some words can benefit with more avaraging than others.)

Be aware of how you release final plosives like "t" and "p". It is often optional in ordinary speech to release such a plosive or not to release it. Consider, for example, the final "t" in the word "eight". You may not even pronounce the "t", or you may replace it with a weak "tah". Or you might put the "t" close to the end of the vowel part of the word, or put it some distance away. Problems can become especially severe when you try to differentiate between words such as "ache" and "ate" (or "eight") and "ape". Use of such similar sounding words in a single vocabulary will give trouble to a human listener as well as to a machine. In general, try to avoid rhyming words with final plosives; these are similar in both vowel and ending parts. (A possible countermeasure is to purposely emphasize the final plosive on one word and not on the other.)

Multi-syllable and acoustically different words give the best results. Consider the telephone operator who pronounces the number "five" in a rather special way so as to differentiate it from "nine", or the pilot who says "zero" instead of "oh", and "niner" instead of "nine". Do not attempt to recognize the letters "c", "d", "e", "b", etc. because they all sound quite alike. Appendix F gives the international phonetic alphabet and also numbers as spoken by pilots and telephone operators.

Generally, the larger the vocabulary and the more similar sounding are the words, the larger will be the error rate. If accuracy is a problem, make use of the sub-groups in the RECOG command. In a two step recognition process, the first recognition can be limited to only a few distinctly different words. Perhaps one of these recognitions then brings up a second group of words which are equally distinctive among themselves. And so on. The way that different word groups can be arranged is almost limitless, partly with the number after the command as RECOG n, and partly by blanking and unblanking certain words in the total template set. And, of course, recognition of a particular word can cause a second complete template set to be loaded from disk memory. A series of recognition steps can in principle involve an almost unlimited number of different words or other sounds, and the same word can appear in different template sets or sub groups in the same set. Programs with menus are well tailored to two step recognition methods, especially "pull-down" menus. A literal example applies to a fast food restaurant where you first select categories such as "sandwich", "drinks", or "deserts". For a second selection you choose the type of drink or the type of sandwich, and then the same word can be used in two or more categories without confusion.

A final suggestion is to make words in the vocabulary all have about the same duration. A word is time normalized to 12 patterns which make up the template. A short word is stretched and a long word is compressed. There can be an impact on

recognition acuracy when word lengths are unequal. For example, a briefly utterred "tea" has the plosive part stretched with the result tending to become somewhat like the word "sh-eee". Or a word like "Commodore", if compressed, will shorten the leading "c" as well as the voiced plosive "d" to be a little more like "t". Time normalization is carried out in order to facilitate uniform (and simple) template comparisons. But it can distort the nature of sounds, especially the fast changing parts of these sounds. The word recognition algorithm prevents the matching of templates if word durations differ excessively, but some differences must be allowed.

The PLAY Program

The PLAY program is written entirely in machine language. It does not employ wedges. In place of these, the BASIC program must direct operation to specific memory locations, which is done with SYS commands. The PLAY program has no means for saving vocabularies or performing word recognition. It is only for playback, including loading vocabularies from disk or tape. It is short and efficient--only two blocks long (256 bytes per block).

From the keyboard, load as

```
LOAD"PLAY",8,1
NEW
```

Note that NEW has been used so as to set pointers for BASIC statements subsequently typed or loaded from disk. This is the same as when loading "VM" from the keyboard, but in this case a SYS command is not required.

The next task is to load in a vocabulary with SYS as

```
SYS 49155 "ENGLISH",8
```

where SYS replaces the FIND wedged command. At this point, a BASIC program can be written and RUN. The PLAY program and the vocabulary remain in memory.

Loading from BASIC statements is done similarly to that with VM as follows:

```
...
150 IF A=0 THEN A=1:LOAD"PLAY",8,1
...
200 SYS 49155"ENGLISH",8
```

The following replacements for wedged commands apply:

FIND"filename",8	to	SYS 49155"filename",8
SPEAK n	to	POKE 151,n:SYS 49152
VOLUME m	to	POKE 253,15-m
SPEED q	to	POKE BASE+266,q
SCREEN	to	POKE BASE+264,0
SCREEN n	to	POKE BASE+264,n

The BASE address refers to the page number where a speech vocabulary begins. The first few hundred bytes in the vocabulary specify various memory locations and other data, followed by the digitized speech itself. The page number is stored in location 56 in page zero. You can compute the BASE address as BASE=PEEK(56)*256.

Location 49155, which replaces FIND when used with SYS, is within the machine language program which starts at location 49152. Each equivalent SPEAK starts the program at this location. The POKE equivalents above which do not involve SYS can also be used with the wedged version, VM, as an alternative.

Location 151 (in page zero) contains the index number of the word to be spoken (range 0-63). Location 253 contains the volume setting, but in a form as 15-v where v represents the number used with the wedged-in command.

When a recording is made, SPEED, SCREEN, and RATE are contained within the vocabulary itself. Parameters may or may not be default values. With keyboard commands or BASIC statements, SPEED, VOLUME, and SCREEN can be changed as desired prior to playback. But it is not easy to change where in memory the vocabulary was placed when originally recorded.

The wedged command SPEED q is replaced with a POKE having a different number. A table of numbers with corresponding playback samples per second is given in Appendix E.

A word vocabulary can start at any page in the range 16-160. The starting number is used when the recording is originally made with CLEAR n where n is the number. (See section on recording.) If no number is specified, the last specified value applies. At page 16, maximum memory is available for a vocabulary, but that available for the BASIC program is a minimum. At page 160, speech is stored "under" or "behind" the BASIC ROM and will not interfere with the longest possible BASIC program. However, only 4200 bytes are available for speech, which is about 4 seconds, which is enough for perhaps 8 one-half-second words. (Note: Examining speech under ROM requires a special machine language program because BASIC must be turned off in order to examine the data.) See Appendix E for information on memory locations.

Amplitude Editor

The quality and intelligibility of recorded speech can be improved with either of two special BASIC programs called EDITOR.BAS and EDITOR. Each program, written in BASIC and compiled BASIC, respectively, loads in a short machine language routine from DATA statements within the program itself. The compiled version runs considerably faster (cursor movements) but cannot be listed; nor will it function with vocabularies that start at a lower page number than 64.

With VM in main memory, type

```
LOAD"EDITOR.BAS",8   (Standard BASIC)
LOAD"EDITOR",8       (Compiled BASIC)
```

and RUN. Another way to load and run is to select the EDITOR option from the main MENU. (Note: We will henceforth refer to both versions of the editor as simply EDITOR.)

There are two ways to get words into memory for editing. Words may be recorded one by one while in the edit mode, or a previously recorded vocabulary can be loaded from disk. In either case, the final result can be saved back to disk memory. The EDITOR program presents a menu from which the appropriate selection can be made.

VM Jr. converts speech to a rectangular wave which is sampled (at the specified RATE) and placed in memory as "1's" and "0's", usually several of each in sequence. Speech is played back by reversing the process. Loudness of the played back result is controlled in the Commodore "SID" chip which provides 16 amplitude levels including zero. Except for sensing word start and stop, the VM Jr. encodes all amplitudes at the maximum value. (The standard Voice Master encodes amplitudes as well.) The coded speech in memory shows a byte in which 4 bits specify amplitude of the following 12 bytes of fast samples, then another amplitude byte, 12 more bytes of fast samples, and so on. The edit display shows amplitude levels throughout the word (all maximum initially) with these levels individually controllable downward. Whether or not amplitudes have been modified or even zeroed, the 12 bytes of fast sampled data remain and the original signal can thus be recovered. The number of bytes between amplitude samples can be changed by software, and the number will vary with the RATE. (See Appendix E.) The value 12 applies to the default RATE of 7.

The EDITOR program shows the amplitude levels throughout the word in convenient graphical form, with cursors to keep track of where you are in the edit process. From the menu for EDITOR, select the function key F2 to LEARN a current word, with a chosen index number for the word. Then record the word and edit it. Or select number 1 to load a speech file, then type in the file name, then proceed to edit specific numbered words.

To edit the speech, select the function key F1. The speech amplitude data then appears on the screen. Edit with 3 pairs of keys. The first pair, "," and "." (or "<" and ">" without shift) will move the edit window left or right. The four cursor keys used in normal screen editing are also used with EDITOR. The left-right cursor keys move the edit cursor to select individual amplitude bytes for editing. The up-down cursor keys are used to raise and lower the editing cursor. Once positioned as desired, press the space bar to set the amplitude value at the new higher (louder) or lower (softer) value. Lowering amplitude to zero turns off that particular group of 12 bytes of sound.

You can hear what the edited speech sounds like. Select the SPEAK CURRENT WORD option from the menu, or function key F1 while still in the edit mode. The entire word plays back. You can play back the word starting from the left side of the screen to the edit cursor with F3 when in the edit mode. The "," and "." keys in combination with the edit cursor provide control for words that are too long to fit the screen.

Not much more can be said about the mechanics of editing. Gaining practical experience is more valuable. Try recording a word such as "six". Reduce amplitude of the beginning "s" part and see if it improves the word. Do the same for the ending "s" sound. Next try reducing amplitudes following the end of the voiced "i" sound so as to enhance the sudden amplitude drop. The word might then be a little easier to understand.

But you can do more. Try changing "six" to "ticks" by putting an amplitude gap just before the voiced "i" sound and by shortening the leading "s" sound, but not weakening it. Try making the "six" into "sick" by eliminating the final sibilant "s". Your objective is to gain skill in improving words and changing them as you wish. And you will learn quite a bit about the nature of speech itself.

You can leave the edit mode and return to BASIC (instead of MENU) with RUN/STOP and RESTORE.

The edit program does not directly allow beginning and ending parts of words to be deleted. Such a procedure could in many cases shorten the words so that less memory would be required for storage. This manipulation is possible by modifying memory locations for words with suitable PEEK(s) and POKE(s), but the process is not simple. A better procedure is to use the more extensive "Speech Construction Set" with the standard Voice Master (not VM Jr. which does not measure voice fundamental pitch). With this program, words can be shortened, even during a prolonged sound, voice pitch can be changed, and pitch periods can be repeated to achieve noise reduction. An extremely versatile capability for creating and changing words is provided by the "Speech Construction Set".

Those who want to directly experiment with speech data files can do so with the aid of memory location information in Appendix E.

MUSIC PROGRAM: TRACKER

There are two music related programs, one which is for composing with ability to save and play back your composition, and another which is strictly for live performance--a kind of "electronic kazoo". The two programs are separate and distinct from one another and also from speech recording/playback and recognition. In this section, the "kazoo" is described; the name of the program is TRACKER. The more sophisticated and complex COMPOSER will then be explained. A detailed understanding of TRACKER is not a prerequisite for working with COMPOSER.

Either program expects you to whistle to the computer or play an instrument or even input tones from a simple toy keyboard. In TRACKER, a toy organ can become a synthesizer. The Computer "tracks" tone frequency and makes various transformations and computations to yield the full range of capabilities of the sound interface device (SID) in the computer. The TRACKER program does not permit you to save music to disk or digital tape for later playback through the computer. Use COMPOSER if this is your intent. Music made with TRACKER can be played to an audio tape recorder and saved in this way.

The principal program, TRACKER, is written in BASIC. When RUN, the program calls a machine language routine which does most of the work. The machine language program is called PERFORMANCE.O (O = capital letter). There also is a sequential file called "PRESET-1" which contains previously prepared sounds for use with PERFORMANCE. A complete backup requires all three programs.

Load the program in one of two ways as follows: From the Main MENU for VM Jr., select item number 7 for TRACKER. Or if you are in BASIC, or working with a disk containing only music programs, then LOAD"TRACKER",8 and RUN. After a few moments, you will be advised that presets are to be loaded (a file called "PRESET-1") from cassette (T) or disk (D). Press RETURN for disk. After the presets are loaded, a menu with 16 numbered choices will appear. The following listing describes all 16.

1-8 = PRESETS. (Select one.) This selection choses one of eight user definable presets. The presets define settings of the Commodore's SID music synthesizer. You can alter them in many ways to obtain a wide variety of sounds. Each group of eight presets can be saved to disk or cassette using selection number 10 (see below). It is suggested that you do not change the original ones supplied on the disk. Each group of 8 presets can have a distinctive file name.

In order to familiarize yourself with a preset, try Option 1, a "funky bass", and start whistling or otherwise entering tones. With just a little practice, you will be amazed at how quickly you improve your musical abilities. As you enter tones, you can change from one preset value to another (within the same group of 8) to produce music with great versatility.

9 = DEFINE (presets). This option allows you to observe and/or change the presets that define the SID settings. You will be asked what preset number you wish to examine. In all cases, current values will be displayed in reverse video on the screen. Pressing the return key without entering new values leaves the current values unchanged. There are seven octave values available. Selecting "4" will play back the same pitch that you entered. There are 12 interval values. Selecting "1" will play in unison with your voice at the selected octave. An interval of "5" will play a major third harmony with your voice. Note: The interval refers to the note on the chromatic

scale and can be identified by counting the black and white keys on a piano keyboard. Example: If you enter a "C", the fifth interval is five notes up (including the "C"), which gives an "E". (Example: C,C#,D,D#,E).

The SID's ADSR function (attack/decay, sustain/release) takes control with the onset of your tonal input. The release cycle starts after you stop producing the note.

Oscillator 1 of the SID is always set to ring modulate with Oscillator 3. (See Commodore manual for definitions and details.) An example of this is preset option 7. In order for ring modulation to be audible, the triangle waveform of Oscillator 1 must be selected and Oscillator 3 must be set to some frequency other than zero. None of the other parameters for Voice 3 has any effect on ring modulation.

10 = SAVE/LOAD/DIR (presets). Use this option to save or load your eight presets to or from disk (or tape cassette). You can also view the disk directory with this option (except for tape systems).

11 = INTEGRATION TIME. This selects the number of input tone cycles to be analyzed before the note is played or displayed. A long integration time will limit the tempo of the music.

12 = CONTINUOUS (Output mode). In this mode, the music is continous such that the output will closely track the input frequency with note values between those designated on a musical scale. This can be especially useful for portamento and vibrato effects.

13 = DISCRETE (Output mode). In this mode, the music playback frequency is that which is closest to a note on the chromatic scale. That is, only twelve notes are allowed per octave, without intermediate tones.

14 = ON (Display). This option will display octave and note values in the lower right corner of the screen. The displayed notes correspond to the closest note in the chromatic scale. Up to three notes are displayed at a time depending upon what SID oscillators are activated and upon the interval and octave settings. Notes are displayed with sharps only.

15 = OFF (Display). This turns off the display for option 14 above and achieves fastest response time. Other speed-up techniques are to have output mode on continuous and use a short integration time.

16 = END. This will terminate the TRACKER program and return you to the VM Jr. main MENU if TRACKER was loaded from this menu. Otherwise, return is to BASIC.

MUSIC PROGRAM: COMPOSER

The COMPOSER program creates music when you whistle or play a tonal instrument such as a recorder or a simple electronic organ. Once created, many changes can be made. For

example, the music can be edited, corrected for off-key notes, rests can be added, individual note durations can be changed, tempo can be manually varied, etc. On playback the versatility of the Commodore SID music circuit is available. The VM Jr. measures the frequency of the input and converts it into musical notation. COMPOSER is a unique and sophisticated program with necessary directions provided in this manual.

(Note: With the standard Voice Master, you can also hum to produce music because an additional circuit is included which extracts fundamental voice pitch. VM Jr. does not contain this particular circuit.)

Loading the Program. Load from the Voice Master main MENU by selecting the menu option COMPOSER. The program can then be used directly. If loaded from BASIC (including when the main MENU program is not on the applications disk you are using), type LOAD"COMPOSER",8,1. Then type SYS 16032 and RUN.

Menu Title Page. Press any key and you are asked to specify the printer you are using. Gemini, Star, Epson, and Panasonic are supported as well as the Commodore brand printer. If you don't have a printer, press RETURN for each question.

Main Menu. After answering printer questions, a menu appears. Move the cursor around with the up/down keys to rest on items in the list of menu options. Press RETURN to select the option. Or you can type the number or letter of any given option and the cursor will move to the proper row. Then press RETURN. (Note: If the program does not respond to the keyboard, check the shift/lock key. It might be down.)

The Learning Process. Select option 1 and RETURN. You should see a red recording screen with bass and treble clefs, some words at the bottom of the screen, and eighth rests scrolling by. Try whistling into the microphone (or play a flute, etc.). A note should appear on the screen. Try whistling higher and lower notes. Hold a note for a time. Notice that as soon as you stop whistling the note scrolls by and the computer waits for you to enter the next note. Try a little tune. Don't make notes too rapidly else the computer might not pick up all of them. Also be sure to put a measureable gap between notes or the computer will think that you have not made up your mind on the first note. (See MODE option.)

At the bottom of the screen is a text window showing the things you can change, plus other data. The computer is only capable of reading the keyboard when there is no sound being detected by VM Jr. Also, it may be necessary to hold down a key until the appropriate response occurs.

List of Options.

MODE. Pressing the "M" key toggles between DISCRETE and CONTINUAL modes. If the mode is DISCRETE, you have to stop whistling (or entering a note) in order for the computer to display the next note. If you select CONTINUAL, eighth notes at the given tempo continue to be plotted.

SOUND. Press the "S" key to toggle the sound on or off. Sound ON means it plays back the displayed note as you whistle (similar to what happens with TRACKER). Sound OFF does as it says. You can always play back in the EDIT mode.

OCTAVE. Press the "O" key (the letter, not the digit) to toggle between NORMAL and DOWN octaves. Octave DOWN means the computer records every note as if you had whistled it in an octave lower than you really did. Octave NORMAL records the actual pitch value.

TEMPO. Pressing the "T" key selects between fast, medium, and slow tempo settings. The tempo determines the playback and recording speeds, that is, the rate at which notes change in eighth note intervals.

NOTE. This indicates the value of the current note displayed between the markers at the extreme right side of the screen. Octave is not specified. Notes are always printed with sharps, never flats. (No problem because A# is Bb, etc.)

VOICE. This indicates the voice setting (1-8) currently in place. Refer to the section on voices. (Similar to presets for TRACKER.)

Return to Menu. After producing a song by whistling and experimenting with tempo, get back to MENU by pressing the space bar. You can also get out by typing "E" which bypasses the main MENU and places you in the EDITOR. The next time you go back to the record mode, the next note will be placed at the end of whatever you have already recorded (unless you erase the buffer).

Playing it Back. Select option 2 from the main MENU and press RETURN. The screen color changes to blue and the music plays back. Press the space bar at any time during playback and return to the menu. You can also press any of the keys that worked in the record mode except the "M" key. Pressing "E" places you directly in the edit mode.

Edit Mode. This is menu option 3. The screen changes to green. There are two important keys to remember, the space bar which returns you to the main MENU, and the left arrow key which displays the help screen. While in the edit mode, the note being edited is the one at the far right on the screen between the markers.

The edit mode supports all of the commands that can be used in the record mode with the exception of mode "M" (no response) and the "E" key (which means something different as described below).

A list of edit commands can be reviewed by pressing the left arrow key while in the edit mode. Press any key and get back to the edit mode. The list of edit commands follows.

```
A.....................Add a note.
B.....................Jump to beginning.
C.....................Change a note.
D.....................Delete a note.
E.....................Jump to end.
Up arrow or I.........Raise note.
Left arrow or J.......Scroll left.
Right arrow or K......Scroll right.
L.....................Alter note duration.
Down arrow or M.......Lower note.
O.....................Octave normal/lower.
P.....................Print composition.
R.....................Insert eighth rest.
S.....................Sound on/off.
T.....................Change tempo.
0-8...................Selects voice.
Space bar.............Leave edit mode.
< or , (comma)........Shorten note duration.
> or . (period).......Lengthen note duration.
/ or ?................Insert a measure bar.
```

These many edit commands are explained in more detail in the following. (Note: The version of COMPOSER for the standard Voice Master has 3 more edit commands, F, H, and W, hich refer to the hum input mode.)

A: Add a note. This waits for you to whistle in a note, and then inserts it in front of the note displayed at the markers.

B: Begin. This places you at the beginning of your composition.

C: Change a note. This waits for you to whistle a note and then replaces the current note with the note you whistled.

D: Delete a note. This deletes the note between the markers from your score. It also deletes rests and measure bars. All the notes to the right of the markers are shifted to the left one position. If there is nothing left to delete, you will be returned to the main MENU.

E: End. This places you at the end of your composition.

Up arrow or letter I: (Press shift key with cursor key.) Raises note one half step in pitch. Each press of the key raises the note an additional half step (holding the key down does this automatically).

Left arrow or letter J: (Press shift key with cursor key.) Moves left one note in the note buffer.

Right arrow or letter K: Moves right one note in the note buffer and also plays back the note. Hold the key down to play back the composition at the selected tempo.

L: Pressing this key first changes the note you are at to an eighth note and then steadily increases its duration at the selected tempo as long as you hold down the key. Release the

key and scroll to the next note. Use the "L" key to tap out the rhythm for a melody.

Down arrow or letter M: Lowers note one half step. (See up arrow.)

O: Shifts octave value lower or normal.

P: Print. This dumps everything on the graphics screen to the printer. Then it moves you ten notes further into the music and you can press "P" again in order to print out the next part of the composition.

R: Rest. This inserts an eighth rest at the current position.

S: Sound on/off toggle previously described.

T: Change tempo previously described.

0-8: Select voice. Typing one of these numbers (0 through 8) changes the sound, or voice, of the current note, as well as all the notes following it, until another sound setting is encountered. A zero clears the sound and one through eight sets the sound. The number for the sound appears just below the note being played. Each note can have a different sound quality if desired. The sound/voice settings are user definable by selecting option A in the main MENU.

Space bar: As indicated.

< or comma: Shortens the duration of the current note.

> or period: Lengthens the duration of the current note.

/ or ?: Inserts a measure bar for "decoration". The measure bar does not affect the way the music is played, but these bars do help to locate parts of a printed piece of music.

Saving and Loading Music. From the main MENU save the composition with option 6. You will be asked for the filename. Chose one and ENTER.

Option 5 is for reloading a previously saved file. There also is an append option which loads a file from disk and appends it to the end of the composition currently in the buffer. This feature is useful for repeating a section of your composition. You can append repeatedly. To use the append feature, save the section to be repeated under its own filename.

Changing Voice. Eight different voices or sounds can be created for selected use in any one composition. The music synthesizer in the Commodore provides three tone osicllators, noise genertors, and envelope modulators. You can define a voice to sound like an explosion or a musical chord. Each voice can be defined at one of 16 different loudness levels which can be varied during playback.

If you select main MENU option A, you are prompted to enter requested information. Current settings are shown in reverse video. To retain a setting unchanged press RETURN without a number. You will first be asked which of the 8 voices you wish to define, numberd 1 to 8. If you enter 0, you will get back to the main MENU. After selecting the particular voice, you will be asked the following for each of the three sound channels:

On? Type yes if you wish this channel to be active.

Waveform. Enter the letter for the type of waveform you wish to be used for this channel: "T" for triangle, "S" for sawtooth, "P" for pulse, "N" for noise.

Attack. Select envelope attack time. Range is 0 to 15.

Decay. Enter envelope decay rate. Range is 0 to 15.

Sustain. Enter envelope sustain level. Range is 0 to 15.

Release. Enter envelope release time. Range is 0 to 15.

Octave. Select octave value for playback. A zero or four plays back the octave originally entered. A value of 1 selects the lowest octave with 7 the highest.

Interval. This is the number of half steps up in pitch (less 1.0) from the note displayed on the screen. Thus, for a C, 1 would mean play C, 2 would mean C#, 3 would be D, 4 would be D#, up to 12 which would be B. Zero is a special case, playing a note slightly out of tune. Playing interval zero in one sound channel with interval 1 in another produces a flanging, or ring modulation, effect. Playing interval 1 with, say, interval 5, would produce a major third chord. Other chords can be generated in a similar fashion. (Of course, a chord requires use of more than one oscillator. The intervals cited above refer to two different voices.)

Pulse Width. These two bytes, a low byte and a high byte (as defined in the Commodore User Manual), are used to determine the width for the pulse. You will only be asked this question if you choose the pulse waveform.

Volume. This is the overall volume for the three channels, with range 0 to 15. (Channel volumes are not separately controllable.)

Filter. Answer "Y" or "N". A "Y" will turn on the SID filter for this channel and the following questions related to filter settings will be asked:

Resonance. This is the amount by which you want to emphasize the cutoff frequency. Range is 0 to 15.

Cutoff Frequency. This is the upper 8 bits of the filter cutoff frequency. Range is 0 to 255.

Lowpass. Answer "Y" if you want a low pass filter.

Bandpass. Answer "Y" if you want a band pass filter.

Highpass. Answer "Y" for a high pass filter.

Changing Key. The options KEY UP and KEY DOWN on the main MENU move all of the notes in the music buffer one half step up or down. When you play back your music, it will be in a different key. If you do a key up or key down when a note is at the very top or bottom of the scale, it will wrap around such that a high note becomes very low and vice versa.

Clearing Memory. Option 4 on the main MENU erases the song you have in memory so you can enter a new one. It will first ask if you are sure you want to erase the song.

Listing Disk Directory. Menu option D displays the directory of your disk. Press any key in order to return to the main MENU. (Not applicable for tape cassette systems.)

Leaving the Composer. Option zero from the main MENU exits COMPOSER with a "warm start" with return to BASIC or the main MENU for VM Jr. Any music left in the note buffer is lost, along with any left-over BASIC programs you may have had in memory before loading in COMPOSER.

Tips on Using COMPOSER. VM Jr. measures the frequency of your whistle or other input tone. It then tries to match that frequency to the nearest note in the chromatic scale. You might tend to produce notes that are consistently a little too high or too low. The composer tends to correct the error if within a one half note interval. But you can correct your own mistakes by watching where the note is printed on the screen. You will be able to correct your accuracy in frequency control quite rapidly. Most users find that they can enter their favorite songs into the computer very quickly and naturally, even if they have had little or no musical training.

Try entering your music at the slow tempo setting and set the mode to DISCRETE. This will result in only eighth notes being plotted, and few if any rests. After you have recorded a short section, go into the edit mode (press "E" from the record mode). Then press"B" to get to the beginning of your composition. Now "single-key-play" your score using the right arrow. When a sour note is encountered, press the "C" key and whistle the proper note. Or if you prefer, raise or lower the note using the up/down arrow keys. You can also use the "A" key to add in a note that was not recorded (or "D" to delete an "accident"). Then continue using the single-key-play technique until you have reached the end of your composition.

Now that the notes are correct in pitch, go back to your starting point, set the tempo to medium or fast, and tap out the rhythm (literally!) using the "L" key. Once this is accomplished, go over your composition and make adjustments to durations using the right or left cursor keys, insert rests as needed, and add in the voices. If your composition repeats a section or theme, consider saving this part and appending it to your working buffer with the append option.

It is possible to use a simple electronic organ in place of a whistle or other musical instrument. You can play the organ to the microphone. Or you might try connecting it

electrically with a microphone cable. However, if notes are low in fequency (below the lowest note you can whistle, which is around 600 hertz or cycles per second), results may not be as desired because harmonics of the note may be mistaken for the note itself. Some simple electronic organs produce square waves which have particularly strong third harmonics. The typical child's electronic organ may cover a range of about 200 to 900 hertz. In order to use such a device, it may be necessary to filter the output so that the third harmonic component is weakened in comparison with the second harmonic. This same problem occurs with the TRACKER program.

APPENDIX A: BACKUP

The Voice Master disk is write protected--the jacket may not be notched, or a write protect tab may cover the notch. It is write protected for the benefit of the user, and not because copying is discouraged. To the contrary, it is suggested that you make at least one copy. You could of course make or open a notch and then record to the Voice Master disk. But then a possible accident could ruin your day!

There are many copy programs on the market, most of which will work well to make a full disk copy or make copies of individual programs. BASIC programs copy easily with LOAD-SAVE sequences (load from the Voice Master disk and save to a formatted disk). Vocabularies can be loaded with FIND and saved to another disk with PUT, and similarly for recognition with TFIND and TPUT. Copying machine language programs is not quite so straightforward but can be done with some third party software. The main wedged program on the Voice Master disk has parts located under the BASIC ROM, which causes difficulty with some copy programs.

If you are limited to software provided by Commodore with the Model 1541 disk drive, it is suggested that you make a copy of the entire Voice master disk. If you do not wish to retain all of the programs and files, then delete the unwanted ones. Start with a clean newly formatted disk. To format, use the Commodore keyboard command PRINT#15,"NEW:name,id", preceded with OPEN 15,8,15, and terminated with CLOSE 15, all as described in the manual for the disk drive.

Utility programs that come with model 1541 disk drive include two of interest here. If you have two drives (device numbers 8 and 9), then you can use "UNI-COPY". If you have only one drive, then use "SD.BACKUP.C64" and go through a sequence of swaps between the Voice Master disk (source), and the destination disk (newly formatted). These programs automatically take care of opening and closing files.

Individual files and programs are eliminated with PRINT#15,"SO:name". The manual for the disk drive suggests "SCRATCH:name", or "S:name" for short. (There may be an occasional "bug" in the software which can be avoided by inserting the zero (0) after S as written first above.) Be sure to open the file before using this procedure, and close it when you are finished.

A useful disk for handling Voice Master commands must include at least VM and/or PLAY, where only PLAY need be retained if your interest is limited to playback. The program BAR might also be included in order to check on general microphone usage and input sufficiency if speech recordings are to be made or speech word recognition is to be achieved. Vocabularies can be loaded from a disk using only PLAY, but making a vocabulary (or doing word recognition) requires VM.

A disk copy of individual machine language programs uses the program BACKUP, written in BASIC. This program calls a machine language program contained within BACKUP as DATA statements. The Voice Master disk contains machine language programs as follow:

VM	Main program with wedges.
PLAY	Playback program without wedges.
BAR	Bar graph.
PERFORMANCE.Ø	Part of TRACKER for music. (Letter O.)
COMPOSER	Main music composing program.

Simply LOAD"BACKUP",8 and RUN. Follow instructions with respect to a source disk (Voice Master disk) and a destination disk (your formatted disk). As previously stated, the minimum requirement for speech is VM or PLAY. For music, have COMPOSER or TRACKER (which also requires PERFORMANCE.O), or have both. Include BAR if it could be of value. It is a quite short machine language program.

APPENDIX B: CASSETTE SYSTEM

In some cases, a tape cassette system represents a more economical choice than does a disk system. Although loading can take considerable time, the process can be quite reliable. For two years, telephone demonstrations of speech, using DEMO, have been given by COVOX with tape software loaded in each morning. No failures have occurred with the tape system and the original tape continues to be employed.

Advantages of a tape system are low apparatus cost and almost zero standby power. (Surprisingly, in terms of cost per unit storage, floppy disks cost less than tapes!) Tapes have a major disadvantage, especially in programs with menus which call up other programs. The main MENU program on the disk is not suitable for tape.

In making a tape based system, it is important that the first program be either VM or PLAY if following programs involve speech input/output. (One could have BAR as a first program because BAR does not use Voice Master commands. But the following program would then have to be either PLAY or VM.) Of course, one can have multiple cassettes with one program on each and swap cassettes as desired. Even MENU might be implemented in this case. But we will consider the more economical procedure here.

The program BACKUP on the disk is used to copy a machine language program (VM, PLAY, BAR, and music programs) from

one disk to another, from disk to tape, or from tape to disk. To make a tape of, say, VM, first LOAD"BACKUP",8 from disk. Then RUN. Simply follow menu dirctions. The name VM is given to the program on the tape. The BACKUP program automatically puts the proper device number in memory location 186 (8 for disk, 1 for tape) so you don't have to do this.

To load VM from tape, put the tape in the cassette player and type LOAD"VM",1,1. Your are next instructed to "PRESS PLAY ON TAPE". You are informed when the program VM has been found. Then press the Commodore key, CTRL, or space bar to complete the loading process. You are now ready to record vocabularies or make recognition templates or produce applications programs.

You will probably want to save a newly created vocabulary or template set from main memory to cassette. You will of course need to have VM in main memory as well (not PLAY which can only load a vocabulary from cassette or disk). Have the tape wound past VM and type PUT"vocabulary",1 in order to save a vocabulary, or type TPUT"template",1 if you have a template set to save. Then activate the recorder following prompts as before.

When you reload everything from tape, and after VM is in memory, type FIND"vocabulary",1 or TFIND"template",1

You could load a BASIC program after VM and before FIND and then let your program automatically load the vocabulary. Save a BASIC program to tape after loading from disk or just after writing it on the keyboard with the usual SAVE"name",1 with prompts as before. Always remember, however, to put a "reasonable" length of blank tape between programs and be sure that loading starts from one of these blank places. To load your BASIC program into main memory, just type LOAD"name",1.

You may wish to make a tape of, for example, the demonstration program CALCULATOR. This requires a tape with VM first, then the BASIC program for CALCULATOR, and then the vocabulary "ENGLISH" followed by "SPANISH" (or your own). On loading back from tape, the BASIC program automatically loads in the language once you have selected it. If you have both "ENGLISH" and "SPANISH", then the proper program will be loaded, even if another program is first skipped. This illustrates a situation that can become cumbersome. Suppose the desired vocabulary is far down the line--near the end of a C60 tape. You may have to wait almost a half an hour! Having many short tapes appears to be preferable.

For quick demonstrations, DEMO is handy. First put VM on the tape. Then the BASIC DEMO program. And finally, the vocabulary called "GREETING" which says "Covox Voice Master-ter-ter-ter".

Important note: Demonstration programs have been written primarily for use with a disk drive. Upon exiting from a program sequence, the program may be asked to load some different program. In the case of DEMO, it seeks to load MENU,

which is too awkward for normal use with cassettes. You may wish to revise the program to avoid a situation that might require you to restart the computer. Or, just be careful and exit with RUN/STOP, perhaps with RESTORE as well.

You will probably also want to have the BASIC program EDITOR on tape. This program contains a machine language program included as DATA statements. Thus you need not use BACKUP, but only the normal load/save procedure as for any BASIC program. A similar situation exists with BACKUP itself, which calls forth a short machine language program contained in the BASIC program as DATA statements.

It is suggested that you make an entirely separate tape for music programs because these have no relationship to VM or PLAY. (CALIB is not available for music programs. Use the VM Jr. level light for this purpose.) You may wish to have BAR as well in order check levels. COMPOSER is a single rather lengthy stand-alone machine language program. Use BACKUP. The BASIC TRACKER program on tape loads in both the machine language program PERFORMANCE.O as well as the BASIC-type file "PRESET-1". (The O is the letter and not zero. It stands for object code.) On a tape system, you will load in TRACKER as any normal BASIC program. Then when you RUN it, the other two programs will automatically be loaded in. The order with which they appear on the tape must follow the order of loading in the BASIC program.

When a BASIC program automatically loads in a second machine language program from disk, the device number is set at 8. Voice Master demonstration programs use the variable "D" to designate either disk or cassette. The program PEEK(s) memory location 186 to get the current device number and then this number is used to accomplish the desired device load or save. For example:

```
D=PEEK(186)
LOAD"PERFORMANCE.O",D,1
```

Location 186 contains the number 8 if loading is from disk (or 9 if a second disk) or 1 if from cassette. If PERFORMANCE.O is loaded from disk to main memory, then saved to cassette, and subsequently loaded from cassette to main memory, the original value of 8 for D becomes 1.

Finally, how can you make a backup copy of a tape? Here is a case where one must be practical. Simply use a radio or tape player with two cassette drives. Or have two players, one to play and the other to record at the same time. It is much simpler than going through the backup process program by program.

APPENDIX C: AUDIO ALTERNATIVE

The most commonly used video display device is a television receiver. This attaches with an RCA type plug. A modulator in the computer changes visual and auditory data to a high frequency signal just like a TV broadcast station.

Some computers are equipped with a monitor which is similar to a TV receiver except that the signals it must receive are not placed on a radio frequency carrier. Audio and video are on separate wires. Most such monitors have considerably better resolution than a TV receiver, with superior graphics and ability to display 80 columns of text. The Commodore provides a 5 pin "DIN" connector on the back of the case to which a monitor video system can be attached. (A TV receiver monitor will function at the same time as a video monitor.) The standard cable used for this purpose has two (or three) RCA type connectors at the end, one (or two) for video and one for audio. A monitor must always have a socket for at least one video line. But some monitors do not have audio. You must then acquire a separate audio amplifier and speaker. The audio direct from the computer does not have much power, but may be enough for earphones.

Whether or not your video system has audio, you might want to send audio someplace else, such as a radio transmitter or a high power amplifier. You can get audio from the SID chip from the DIN cable mentioned above. Or you can attach audio direct to the 5 pin DIN connector. (Pin No. 2 is ground and pin No. 3 is audio.) You could send audio over your intercom system. Or put it to a wireless broadcaster so that FM or AM receivers can pick it up to a range of one or two hundred feet. Or record on an audio cassette player. Or switch audio from one place to another, such as from one classroom student to another.

Capabilities of the Commodore sound interface device (SID) are used for audio. For speech, only the digital to analog part of the SID is actually employed so that speech loudness can be set to any one of 16 levels, including zero. If it were not for the SID chip in the Commodore, a separate 4 bit D/A converter would be required if other than constant level speech or music were desired. Using the SID chip has the advantage of permitting the programmer to handle speech, music, and sound effects in a single audio channel.

APPENDIX D: COMMAND SUMMARY.

Recording and Playback

SPEAK n. Designated word or phrase in the range 0 to 63. Plays back through the TV monitor or external amplifier. SPEAK(ing) a phrase that does not exist gives a tone beep. The space bar resets to the start of the word being produced.

LEARN n. Word or phrase with index n is recorded in main memory. Re-entered phrase replaces previous one. Stop in-process recording with any key. When waiting for input, put on hold with the large up-arrow key, and press a second time to return to LEARN. Too long an input state causes time-out.

PUT"filename",8. Saves vocabulary on disk, starting on page n (see CLEAR). Also saved are speed, volume, and screen settings.

FIND"filename",8. Recovers named vocabulary previously PUT. Retains same starting page address and speed, volume, and screen settings.

CLEAR n. Number n either omitted or given in the range 16-160. Clears vocabulary from main memory and prepares for introducing a new vocabulary from the microphone. Sets parameters to normal (default) values. The number n specifies the page in memory where the vocabulary starts (decimal 256*n). With n left unstated, the default value of 64 is inserted. Loading a vocabulary sets page number for the original recording and CLEAR(s) any vocabulary previously in main memory. The number n applicable to a stored vocabulary cannot be changed.

SPEED n. Changes playback speed in the range 0-10 with normal (default) value of 7. Speed of playback proportional to n.

VOLUME n. Changes output volume. There are 16 levels, 0 to 15, with 15 the loudest and 0 being zero. Amplitudes greater than 15 are limited to 15. Normal (default) value is 15.

PAUSE n. This acts like a software timing loop and produces a fixed delay. n is the number of 1/4 second delay increments.

SCREEN n. During speech playback, the normal (default) condition is for the video screen to remain on. Screen n (n a positive integer) blanks screen on playback. Return to default condition with SCREEN (no number).

RATE n. Changes the sampling rate at which the speech is digitized. The range is 0 to 10 with the normal (default) value 7. A nonstandard recording RATE will not play back at a normal speaking rate unless SPEED has the same index number.

CALIB. Displays question mark (?) as in LEARN and TRAIN. Does not actually record. Subject to time-out. Use for checking amplitude range and especially zero levels when not inputting sounds to the microphone. Leave CALIB with any key.

Recognition

TRAIN n. Designated word in the range 0 to 31. Re-training the same number creates an average template. Terminate a TRAIN with any key. Up-arrow puts TRAIN on hold. Time-out if word not produced in time or if too long.

BLANK n. Clears the template for word n for n in the range 0-31. BLANK without a number clears all templates. BLANK a word before TRAIN in order to avoid averaging.

UNBLANK n. Recovers the template previously BLANK(ed), or all templates if index number n not used.

RECOG n. Program waits for an input and attempts recognition by comparing the template made for the input word to those in memory. If no number is specified, all 32 templates are scanned. For n=1,2,3, and 4, template numbers 0-7, 8-15,

16-23, and 24-31 are scanned respectively. RECOG 3,1 results in scanning groups 3 and 1 in that order only. The best fit template number is placed in memory location 151. Other numbers for certain errors and other conditions. Terminate with any key. Time-out is applicable.

ACCEPT n. Index n in range 0-4 sets error criteria. Zero value accepts all words. Value 4 has close tolerances. Variations in error measures can be changed in memory locations.

TPUT"filename",8. Saves a set of 32 templates in main memory to disk memory.

TFIND"filename",8. Transfers a set of 32 previously saved templates from disk to main memory.

APPENDIX E: MEMORY DATA AND LOCATIONS

Programs which manipulate speech recording, playback, and recognition utilize several addresses in page zero of computer main memory. Many of these locations must be retained such that they are not normally available for user written machine language programs. However, contents of these memory locations can be stored and replaced as part of a user program.

Three areas of memory contain various VOICE MASTER programs. A major portion of the machine language program is contained in a 4K memory block between BASIC and kernal ROM's. A general work area for speech recording is contained behind the kernal. Some numerical data for recognition and templates are located behind the BASIC ROM. Stored vocabulary words for playback are located as desired through use of CLEAR n.

It must be realized that the memory map for a computer is quite specific to that computer. Voice Master programs are not transferrable from one computer to another without numerous adjustments being made, even between computers having the same type of microprocessor found in the Commodore 64. Machine language programs, especially for speech recording and playback and disk (or tape) storage and retrieval tasks, make frequent use of utility programs in the kernal.

Many of the memory locations in this section refer to a "BASE" address, which is defined in the CLEAR statement used in conjunction with making and saving a vocabulary.

Memory location 3 ($3 hex). Contains parameter setting the time-out value equal to the number of half-second intervals.

Memory location 56 ($38 hex). Contains the page number of memory where speech data begins.

Memory location 151 ($97 hex). Current phrase number recorded or spoken during speech recording and playback with a range 0-63. In recognition, contains index number of best match, or a number defining error or other parameter related to recording or recognition.

Memory location 253 ($FD hex). Volume setting to be subtracted from 15. Range 0-15.

Memory locations BASE to BASE+255. These 256 bytes of memory define starting and ending addresses of where a recorded phrase is stored in memory. For example, to find where the 7th phrase is stored, compute 7*4 plus BASE. PEEK this memory location for the low order byte of the start of phrase 7. The high order byte follows. The next two locations are the low and high order bytes, respectively, for the ending address for phrase 7. (Note: PEEK will not work if CLEAR has used the numbers 159 or 160. The leading page of information must not lie under BASIC ROM else needed data cannot be acquired.)

Memory Locations BASE+256 and BASE+257. These two memory locations define the current top of speech memory.

Memory location BASE+259. Total number of recorded phrases. Range is 0-63.

Memory location BASE+264. Screen flag. Setting to a non-zero value will turn off the screen during speech playback.

Memory location BASE+265. Recording rate setting. Same numbers as for SPEED setting.

Memory location BASE+266. Playback SPEED setting. Same as in recording RATE setting. A table of settings for SPEED follows. With POKE(s) instead of the wedged-in command, any number in the range can be used to get intermediate SPEED (or RATE) values. (In this way, a few musical notes can be expanded to create a keyboard.)

SPEED	Base+266	Sample Rate (hertz)
0	32	4000
1	48	4400
2	64	5000
3	80	5300
4	96	5900
5	112	6500
6	128	7200
7 (default)	132	8000
8	160	8900
9	176	10,900
10	192	12,500

Memory locations BASE+267 to BASE+330. List of 64 bytes giving the order in which the phrases where LEARN(ed). Example: LEARN phrases 3,8,12,45, and 4, in that order, then the memory location starting at BASE+267 will contain 3, followed by 8, then 12, and so on. Memory location BASE+259 contains the total number of phrases LEARN(ed).

Memory locations BASE+331 to 45567. This is where the actual digitized speech is stored. For the minimum value BASE=16, compute the total available memory for speech as: 45312-16*256=41216 bytes.

Memory locations 49155 to 49164. First part of memory in which VM resides contains numbers and parameters relating to error criteria in recognition. See section on "Error Criteria, Thresholds, and Hints".

Memory location 49165. Number determines shortest phrase that can be recorded. Normally set to 10. If set too short, the recording can start from clicks or low level background noise. If too long, initial parts of speech can be missed.

Memory location 49166. Number determines duration of low level sounds at the end of a word which terminates the recording. If too short, the recording can stop at a short time gap when not intended. If too long, delays occur.

Miscellaneous Locations. The BASIC ROM uses locations 40960-49156 (A000-BFFF hex). There is RAM behind this ROM which can be utilized if the BASIC ROM is switched off. But if it is off, then BASIC cannot be used to PEEK or POKE memory locations. The program VM occupies 49152-53247 (C000-CFFF hex) with recognition templates and some other parameters under the BASIC ROM in the range B100-BFFF hex. A speech vocabulary starts at 256*n (BASE) where n is page number used in CLEAR and can extend under BASIC ROM to B0FF hex. The input buffer used to store speech prior to organizing it into a vocabulary of words or templates is located at and above E100 hex, which is under the kernal ROM.

Organization of Vocabulary. Data to the computer from Voice Master consists of a single square wave which follows the rapid changes of the detailed speech waveform with components to thousands of hertz. This square wave is sampled at the RATE n, which is 8000 samples per second for the normal or default condition with n = 7. (RATE 6,7, or 8 will usually be acceptable. Intermediate values are obtainable with POKE.) Samples are formed as sequences of 1's and 0's, usually with several of each type in a sequence. These samples are formed into a series of 8 bit groups, or bytes. But before each group of 12 bytes, a single byte is inserted to indicate average amplitude. All amplitudes above zero are actually set to the maximum value during recording or recognition. (In the standard Voice Master, amplitudes are separately measured in the external hardware and recorded in the amplitude bytes.) In reproducing the speech without any attempt at editing, the amplitude byte remains at maximum and the speech is produced at maximum loudness. The amplitude editor allows each amplitude byte to be reset and this in turn sets the gain of the SID chip in the Commodore. The fast samples create the original square wave, but with a controlled amplitude. Some errors occur because samples do not exactly line up with original square wave edges. This error decreases as the RATE value increases (but a high RATE requires more memory for storage).

The beginning byte of a vocabulary consisting of one or more words (up to a total of 64 words) is at BASE+331. The starting address can be computed (in decimal) as

PRINT PEEK(256*n + 331)

where n is the page number used in CLEAR (default value 64, which is BASE upon multiplying by 256). Each word in this vocabulary has starting and ending addresses that are to be found in the range BASE to BASE+255, with the starting address for the first word being that computed above. (Note that two bytes are required for the starting address, and two more for the ending address: 64 words * 4 = 256.) The first byte for each separate word in the vocabulary consists of a number specifying the number of bytes of high speed clipped speech between amplitude bytes. The nominal value is 12, but this can be changed with a POKE. In fact, the number does vary somewhat according to the RATE value selected. After this leading number byte for a word, the plan is repetitious: Amplitude byte, 12 fast sample bytes, amplitude byte, 12 fast sample bytes, amplitude byte, and so on to the end of the particular word. Each word in the vocabulary can be recorded at a different RATE, and each word can have a different number of fast sample bytes between amplitude bytes. The amplitude byte itself uses only 4 of the available 8 bits (the other 4 being available for special coding). Thus amplitude has a range of 16 levels, including zero. The organization of each word is shown graphically in the following:

APPENDIX F: MISCELLANEOUS INFORMATION

Disk Contents:

Name	Type
MENU	BASIC
VM	Machine
PLAY	Machine
BAR	Machine
BACKUP	BASIC
EDITOR.BAS	BASIC
EDITOR	BASIC (COMP)
DEMO	BASIC
GREETING	Speech file
CLOCK	BASIC
CVOICE	Speech file
CALCULATOR	BASIC
ENGLISH	Speech file
SPANISH	Speech file
BLACKJACK	BASIC
DEALER	Speech file
BJVOICE	BASIC
TRACKER	BASIC
PERFORMANCE.O	Machine
PRESET-1	File
COMPOSER	Machine
DEMO-128	BASIC
PLAY-128	Machine

Phonetic Alphabet

Alpha Bravo Charlie Delta Echo Foxtrot Golf
Hotel India Juliette Kilo Lima Mike November
Papa Quebec Romeo Sierra Tango Uniform Victor
Whisky X-ray Yankee Zulu

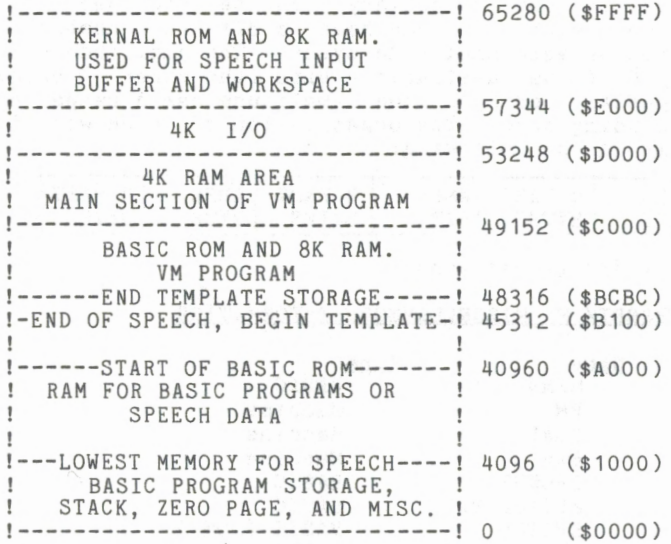
Airman's Numbers

Zero One Two Three Four Five Six Seven
Eight Niner

Telephone Operator's Numbers

Oh One Two Thuh-ree Fow-wer Fie-yuv Six Seven
Eight Nine (or Nie-yun)

APPENDIX G: MEMORY MAP OF C-64 WITH VM



APPENDIX H: SELECTED PROGRAMMING EXAMPLES

You can list the various demonstration programs on the VM Jr. disk in order to study programming methods and techniques. However, these programs have not been written with the objective of making them easy to interpret. (Some may be compiled.) The purpose of this section of the manual is to give some examples that are easier to understand.

Talking Numbers: We will next write a program for a very simple talking keyboard. It will speak numbers 0-9 from "ENGLISH" as you press the numbered keys, followed by RETURN. End the program with a number greater than 9. You must first be sure the VM is in main memory, and then you must FIND"ENGLISH",8 (or "SPANISH" if you prefer) because we are

going to use the spoken digits from this vocabulary. (We use the INPUT statement. The experienced programmer might prefer to use the GET statement and so avoid need to press the RETURN key.) The program is:

```
40 INPUT N
50 IF N>9 THEN 80
60 SPEAK N
70 GOTO 40
80 END
```

Next is a program that speaks numbers in DATA statements (only positive integers in the range 0-9):

```
40 RESTORE
50 READ N
60 IF N<0 THEN 130
70 PAUSE 2
80 SPEAK N
90 GOTO 50
100 DATA 0,5,6,3,1
110 DATA 9,7,8,4,5
120 DATA -1
130 END
```

In this example, we have used a negative number to end the program so that an out of data error statement does not occur.

Now for a slightly more realistic data talking program. We presume numbers are in the range 0-999, positive integers only. This range includes all applications where (positive) decimal values are contained in 8 bit memory cells (sub range 0-255). This program will be even more practical if we set it up to read data from some different program, perhaps one you got from a magazine listing and that you want to check for accuracy by listening to the spoken numbers as you follow along the printed listing with your eyes.

We will presume that the program to be checked, including its DATA statements, does not have statement numbers as high or higher than 10000. We will write our reading and talking program to start at 10000 and then GOTO this number in order to activate the program. You must of course have VM in main memory. Also you must have a suitable vocabulary, such as "ENGLISH". Then you must have the program whose DATA statements are to be checked, or at least the DATA statement part of this program (with line numbers less than 10000). Finally, you must have your special checking program in memory, and this must be appended to the program to be checked. If you know how to append one program to another, then this can be done directly from disk memory. If appending turns out to not be so simple, then there are two avenues open to you. You may enter your program from the keyboard after loading in the program to be tested, or use the "trick" discussed below. (If you have not already loaded in VM from the keyboard, then you must do this loading with BASIC data statements as previously described. Keyboard loading of VM

requires use of NEW, which you certainly don't want to do after loading in a BASIC program or writing one!)

Before proceeding, here is a useful "trick" for appending a short program to a longer one. Write the short program. Keep it short with techniques such as multiple statements per line, avoid spaces and long labels, etc. The objective is to have the entire program appear at once on the video screen with some room to spare. You can save it to disk memory for future use. Whether written directly or loaded from disk, have your short program as low on the screen as possible, and then issue the load command for the program to be tested when the cursor is at the bottom of the screen just below the last statement of your short program. Loading the program to be tested "wipes out" your own program, but it remains on the screen. Now place the cursor on the first statement of the listing for your short program and press RETURN. This will enter that particular statement and combine it with the program to be tested. Continue pressing RETURN until all statements have been entered. You now have appended your program. If your short program is one or two statements too long to permit all statements to be entered in this way, then retyping the one or two lost lines is not too onerous a task. If a "short" program is too long for this procedure to be viable, then repeat the process as many times as needed. Each time you append a segment, re-save the enlarged program, load another segment of the program to be appended, re-load the enlarged program, and enter the added statements with the RETURN key.

If you load VM and "ENGLISH" from the keyboard, then do this first. Otherwise your own program can do this loading as previously described. Next load the program to be tested and write your testing program at the keyboard. Or with the alternative as described above, load in your testing program from disk, then the program to be tested, and then enter your individual testing program statements with RETURN. A suitable program (not compacted for ease of understanding) is the following (where we use FOR-NEXT loops instead of PAUSE for delay between words):

```
10000 RESTORE
10010 READ N
10015 FORJ=1 TO 500:NEXT J
10020 IF N<0 THEN 10200
10030 FOR J=1 TO 100:NEXT J
10040 IF N>9 THEN 10070
10050 SPEAK N
10060 GOTO 10010
10070 IF N>99 THEN 10120
10080 M1=INT(N/10)
10090 SPEAK M1
10100 N=N-10*M1
10110 GOTO 10050
10120 M2=INT(N/100)
10130 SPEAK M2
10140 N=N-100*M2
10150 M3=INT(N/10)
10160 IF M3>0 THEN 10080
```

```
10170 SPEAK 0
10180 DATA -1
10190 GOTO 10010
10200 END
```

The procedure is to break the number into separate integers and SPEAK each by itself. Different program pathways apply depending on the particular mix of integers. The checking program ends when a negative final DATA statement appears. This is guaranteed to occur after all other DATA statements and thus provides a postive ending command. (The reader will recognize that this program could be written with fewer lines by using colons to put two or more statements on a line.)

A second and more efficient program converts each number to a string and then extracts one string element at a time for SPEAK(ing). The example program is:

```
10000 RESTORE
10010 READ N
10015 FOR J=1 TO 400:NEXTJ
10020 IF N<0 THEN 10110
10030 K$=STR$(N)
10040 FOR J=2 TO LEN(K$)
10050 J$=MID$(K$,J,1)
10060 N=VAL(J$)
10070 SPEAK N
10080 NEXTJ
10090 GOTO 10010
10100 DATA -1
10110 END
```

This program will speak multi-digit numbers up to the point where the form is changed to floating point (9 digits in Commodore). Note also statement 10040 where the first string element to be spoken is the second. Numbers are given a leading space for holding a negative sign, and this space remains as an unprinted character if the number is positive (which may not be the case in other computers). Thus the number 24, for example, will be produced as a string with 3 elements. (Note: The sample program can be imnproved with a special spoken word such as "comma" to designate when each new data element is started.)

Two Approaches to a Talking Keyboard: Define a string as

```
A$ = " ABCDEFGHIJKLMNOPQRSTUVWXYZ.,?!'"
```

Use the Voice Master to create a vocabulary where vocabulary index numbers are 1 for "A", 2 for "B", 3 for "C", and so on. We will use index 0 to represent the space bar, which shows as a leading blank in A$ above, and the word will be "space". Don't confuse vocabulary numbers with the string count, which always starts with 1. By choice, we have started the word vocabulary with zero. Thus expect a J-1 somewhere in the program.

We can input a single character at the keyboard with the GET statement as GET B$. We then scan the long string A$ and count each element from left to right until we get a match. The count number is then used in SPEAK N-1. Also, we can print out the identified B$, adding it to a continuing string so as to show what is typed on the screen while also speaking out the letter.

A short program that speaks the letters as the keys are pressed follows:

```
10 A$=" ABCDEFGHIJKLMNOPQRSTUVWXYZ.,?!'"
20 GET B$:IF B$=""THEN20
30 FOR J =1 TO 32
40 IF MID(A$,J,1)=B$ THEN 60
50 NEXTJ
60 PRINT B$;
70 SPEAK J-1
80 GOTO 20
90 END
```

Note that printing is in sequence as a kind of simplified word processor. This approach can be a little slow, especially for characters near the end of the string. We could speed it up somewhat by creating a vocabulary with the more frequently appearing letters of the alphabet in the first part of the string (as done here with space, which is the most frequently seen "character" of them all).

Another method uses the designated ASCII symbol and reference number. Assume that the letter "A" is typed. We get the number 65 as ASC("A") = 65. If we designate B$ as the string representing the typed symbol using GET as before, ASC(B$) = 65 clearly states that the letter is "A". Now simply subtract 64. We get ASC(B$)-64=1 which produces the voiced "A" with SPEAK 1. An example program segment is:

```
400 GET B$:IF B$=""THEN400
410 PRINT B$;
420 N=ASC(B$)-64
430 SPEAK N
```

(A modified procedure is required for the punctuation marks, as well as space, because the proper number to be subtracted is not 64.)

Of course, any practical talking keyboard must contain a number of features to avoid various typing errors or inconsistencies, avoid most non-printing characters, and so on. Also, if the string length is limited, means for handling a sequence of strings must be provided if continuing text is to be presented on the screen.

The Cash Register Vocabulary: A suggested vocabulary for implementing a talking cash register is given below. The number "73" can be spoken as "seventy" followed by "three". In this way, a relatively small vocabulary can handle a large number range.

Index numbers 0 to 20: Same as the spoken number.

Index	Speak	Index	Speak	Index	Speak
21	30	22	40	23	50
24	60	25	70	26	80
27	90	28	hundred	29	dollars
30	cents	31	amount is	32	and
33	debit	34	dollar	35	thank you

If we were so inclined, the words "thousand", "million", "billion", as well as "exponent", "point" (decimal) and so on could be added while still remaining well below the Voice Master limit of 64.

There are many ways we can say $ 4.95. One is "The amount is four dollars and ninety five cents". At a supermarket check stand, you are more likely to hear only "four ninety five". But there are variations. $ 4.00 will be said as "four dollars". And $ 1.00 will be said as the singular "dollar". $0.15 will be read as "15 cents".

Language Translation: Make a template set of the numbers one through five with indices 1,2,3,4,5. Call it "NBRS". Make a vocabulary of the numbers in Spanish as uno, dos, tres, quatro. Use indices 1,2,3,4. Call it "SNRS". Start by recognizing a number when spoken in English. Then PEEK(151) to speak the number in Spanish. The following program will work, where the English "five" ends the program and various error crieteria numbers return for a better input:

```
10 IF A=0 THEN A=1:LOAD"VM",8
20 SYS 49152
30 TFIND"NBRS",8
40 FIND"SNRS",8
50 RECOG 1
60 A=PEEK(151)
70 IF A>5 THEN 50
80 IF A=5 THEN 110
90 SPEAK A:PAUSE 2
100 GOTO 50
110 END
```

APPENDIX I: PROGRAMMING WITH EXTERNAL EVENTS

The computer can function with a variety of inputs and outputs in addition to the obvious ones of keyboard, video display, disk, tape, and modem. The Commodore 64 is especially convenient in this regard because it is equipped with a complete and buffered 8 bit port which can be used for output only, input only, or mixed with some of the 8 bits designated as inputs and others as outputs, all operating together. The "Commodore 64 Programmer's Reference Guide" describes this in greater detail than here (including memory map data). But perhaps some specific examples will serve to encourage more people to take advantage of the built in Commodore capabilities. The first example here will employ the 8 bit port as an input port measuring switch positions as a sort of burglar alarm. You don't need any extra hardware other than a connector to attach to the user port and wire to go to

ordinary single pole, single throw switches. The second example will presume the port to be an output port and we will explain how you can operate remote relays. We will use simple 12 volt relays so that you won't have to deal with the 120 volt house current. Relays operating at 12 volts DC offer the possibility of operating in an automobile or an airplane. Wouldn't it be exciting to fly an airplane by voice control alone?

But the Commodore has more input/output lines than this. Each joy stick connector provides access to 5 binary lines which can be interpreted as 5 separate bits or as a 5 bit number. These are normally used with games in a variety of ways. They can also serve as output lines, but this requires that interrupts be disabled which is a task for machine language programming and is not appropriate in BASIC (without a SYS). As input lines, they are somewhat slow because they are scanned along with the keyboard. But they can be high speed if interrupts are disabled. Voice Master uses the joy stick binary lines with keyboard interrupts disabled.

There is also the tape cassette port. This has 3 binary lines attached to it. Two of these can be high speed input and output lines, and one can be a lower speed output line (the line normally used to turn on and off the tape player motor). These bits come from a very special set of pins on the microprocessor, unique to the model used with the Commodore 64. They represent information in locations 0 and 1 in page zero. Using them requires some attention to the other bits of this port because they are used to turn on and off BASIC and kernal ROMs so as to communicate with RAMs at the same addresses.

We will be concerned primarily with the 8 bit user port. Two memory locations are important. One contains 8 1's and 0's which determine which of the corresponding lines are to serve as inputs (0's) and which are outputs (1's). We can configure the 8 lines with a POKE. But because in BASIC we deal with decimal numbers, the bit pattern that is desired must be replaced with the decimal equivalent. If we want all lines to be inputs, we use

```
POKE 56579,0
```

which is the equivalent of the binary number 00000000. If we want all 8 lines to be outputs, we use

```
POKE 56579,255
```

which is equivalent to the binary number 11111111.

Here is another example. If we want inputs and outputs to follow the bit pattern 00111001, then we would POKE the number 57 into location 56579. (57 = 2↑0+2↑3+2↑4+2↑5, where ↑ means "power of".)

Another location, 56577, contains the number that will be read with a PEEK or sent to a remote controller with a POKE.

Various forms of "masking" can take care of situations where there are mixed inputs and outputs.

For inputs in particular, especially when all 8 bits are inputs, it is possible to read analog variables with up to 256 values--like a temperature or pressure or speed, etc. Or we can multiplex to get many more than 8 different switch values. For example, if we use one of the 8 bits to select between two banks of 7 switches each, we have a total of 14 switch inputs. Likewise, we can use two bits to select between 4 sets of 6 switch inputs for a total of 24. If we don't want to use one of the 8 bits for selection, then there is one bit available from another port, also wired to the user port, for doubling to 16. Or one can adapt the cassette signals or the joy stick signals for this purpose. If we are willing to use two 8 bit numbers to make a reading, one number can address up to 256 different measurement places, and the second number can get the 8 bit value of the reading. One small Commodore 64 can read and log a very large number of measurements indeed!

Talking Alarm System: Use POKE 56579,0 to configure all lines as inputs.

A simple switch to ground at a bit position will result in a zero when the switch is closed or a positive voltage when the switch is open. Eight separate switches will thus reveal 8 separate states or conditions in the external environment. These switches can be normally on or off. A normally on switch can be formed with foil tape on a window such that, if broken, the switch is turned off and the voltage at the bit position to which the switch is attached goes from a zero value to a positive value. (Several windows with continuous foil from one to the next can act as a single switch if foil at any location is broken.) A magnetic switch can tell whether a door or a gate is open or shut. Or if something is moved from a fixed place. A switch can operate from a mat when someone steps on it. A slightly more complex switch can operate if a beam of light is broken or if an ultrasonic signal occurs (or doesn't occur), or infrared radiation, or a sound, etc. Or a switch can tell when the level of liquid in a tank rises or falls past a certain point, or if a motor slows down too much, or if a bottle passes an inspection station on a conveyer belt, and so on without limit.

The computer's task is to monitor these 8 switches. If anything changes from one scan to the next, or from some stored map of how the switches should be set, then the changed switch can readily be identified. The computer can announce the situation in a loud, clear voice, perhaps to a remote speaker or even over radio or telephone.

Monitoring is simplified with masking. This permits individual bits (relating to specific switches) to be examined. A mask is something that selects only certain bits in a byte while setting the remaining bits to zero or 1. The Commodore has a useful capability in this regard (which is not revealed in manuals). Consider the logical numerical example with two decimal numbers as

```
PRINT 57 AND 248
```

Just what does this mean? Nothing--unless the computer turns the numbers into binary form, performs the binary logic, and then returns the binary result as a decimal number. The Commodore does this. The example continues:

```
 57:     00111001
248:     11111000
AND:     00111000    (56)
 OR:     11111001    (249)
```

The Commodore AND and OR statements do this, provided that decimal numbers are in the range 0-255 (commensurate with the limits of a single byte).

Here is a program that speaks out every switch that is closed, i.e., bit value 1:

```
5 POKE 56579,0
10 A=PEEK(56577)
20 FOR J=0 TO 7
30 B=A AND 2↑J
40 IF B=1:THEN SPEAK J:PAUSE 4
50 NEXT J
60 GOTO 10
70 END
```

The number 2↑J (2 to the j-th power) is an 8 bit binary number with all zeros except for 1 in the J-th bit position. AND(ing) this with the actual bit pattern gives 1 only in the J-th bit position. OR(ing) it results in all bits of the original pattern unchanged except that bit J is also 1.

Another approach is to speak out only if a bit does not match some prescribed bit pattern represented by decimal number C. Then change statement 30 above to:

```
30 B = C AND (A AND 2↑J)
```

Voice External Control: The next example is quite practical and can provide voice control of 8 devices with high reliability and accuracy. But first some essentials of logic must be presented.

Configure all 8 bits as outputs using POKE 56579,255. We can POKE 56577,2↑J to turn on the J-th output. But this will zero all the rest of the outputs, which we may not want. Thus we must mask. To do this, first find out the current state of the switches with A=PEEK(56577). Now make a mask of 0's except for 1 in the J-th bit position. Next

```
POKE 56577,A OR 2↑J
```

and we have accomplished what we set out to do. Wherever number 2↑J has a zero, the bit in A will remain unchanged. The bit in the J-th position, only, will become 1. The index J could come from location 151 after RECOG. This logical

operation can serve to turn on a switch with a POKE to location 56577. How about turning it off? As stated, the result of the logical calculation A OR 2↑J is the same as A except that a 1 will always be in the J-th position, whether or not it was there before. A number 255-2↑J will be just the opposite with all 1's except for a zero in the J-th position. Thus the logical result of the calculation B AND (255-2↑J) will be the same as B except that a zero will always be in the J-th position and a switch is turned off with a POKE to location 56577.

Now that we have described the essential logical operations for turning a bit (or switch or machine) on and off, lets describe a practical program which can do the job with high accuracy and reliability. We presume that the operator does not have access to the keyboard and cannot see the video display. He walks around with a headset, speaking commands to a microphone with verifications and advisories applied to the earphone that he wears. He might carry a two way radio so that he is free of trailing wires.

First, he must turn on the complete system. He talks to associates, or to himself. Or he stumbles about or otherwise makes noises. The system must stay off until he is ready. Chances are that a one word activating vocabulary will not ignore casual sounds to the extent desired. He needs an almost perfect padlock. Two different approaches are suggested. In one, he says certain words in a certain order, rather like a combination lock. Or, he can say the same word several times. In either case, the system resets if the sequence is not followed. What kind of errors might we expect? If the probability of error for one word is p, then for n words in perfect sequence the probability of error is p to the n-th power. Example: One word error probability of 0.02 has a three word error probability of 0.000008, or about one chance in 100,000.

In the present example we assume he does not worry about unauthorized activation, so he doesn't need a code. Let him say "begin" three times. Put this word, and only this word, in the sub template set for RECOG 4. Also be tight on accuracy requirements with ACCEPT 2. (As an alternative, you could use BLANK and UNBLANK so that other words in the fourth sub template set could be used as well. But we want to simplify things here). The following program steps do what we want for activating the system.

```
10 A=0
20 ACCEPT 2
30 RECOG 4
40 A=PEEK(151)
50 IFA <> 24 THEN 10:REM RESET
60 A=A+1:REM ACCUMULATE COUNT
70 IF A<3 THEN 30:REM SAY WORD 3 TIMES
80 SPEAK 8:REM "SYSTEM ON" IS WORD NO. 8
```

At this point the system is active. Template 24 is the key word "begin". Vocabulary word number 8 is "system on". The

system cannot go on unless "begin" has been recognized 3 times in a row without intervening errors of any kind.

The operator can reset the system by producing a word other than "begin" in place of where "begin" would normally occur. Also it would be helpful to include another feature, namely, a time-out so that individual words must be spoken within a specified period of time, else a reset is automatic. The Commodore 64 has a built in clock that can be used for this. A timer counts numbers at a rate of 60 per second from the time it is set. Time starts from when the computer is first turned on, or when it is reset. The counter is read with PRINT TI. If seconds are preferred, then PRINT TI/60. A clock in hours, minutes, and seconds is read with Print TI$"hhmmss" measured from when it is reset. It resets to zero with TI$="000000" which also resets the counter TI. (TI$ can be set to anything desired.) Thus if one wants to time something, use an IF-THEN statement following a reset. For example, in the foregoing program, putting a time limit on each of the 3 "begin" words of, say, 5 seconds, results when two statements are inserted in the foregoing program as:

```
35 TI$="000000"
65 IF TI/60 > 5 THEN 10
```

If the operator must say all three words in a specified length of time, then two or even three RECOG statements will be required so that the timer can be set once and not reset each time a repeated "begin" is produced.

The next task is to say whether the operator wants to turn a a switch on, or turn it off. Another sub group of templates is suggested. We will leave the ACCEPT level the same. But before proceeding, lets define template numbers in recognition and word numbers for voiced verification applicable to this particular example.

	RECOGNITION		PLAYBACK	
Template		Word Type	Word No.	Word
1:	0	0 (device)	0	0 (device)
	1	1 "	1	1 "
	2	2 "	2	2 "
	3	3 "	3	3 "
	4	4 "	4	4 "
	5	5 "	5	5 "
	6	6 "	6	6 "
	7	7 "	7	7 "
2:	8	yes	8	the system is on
	9	no	9	say again
	10	reset	10	the command is
3:	16	on	11	OK to act?
	17	turn off	12	the device is
	18	reset	13	on
4:	24	begin	14	off

Brief note on above: You can say the numbers to designate the devices, and have the numbers come back as pre-recorded

words. Or you can choose some other names such as door, window, crane, or whatever. We leave the device naming to your imagination. Also note we do not say "off" and "on", but rather "turn off" and "on". This gives us a little safety factor because "on" and "off" can be difficult to distinguish.

Now back to the task at hand--to turn a switch on or off or reset. A confused word can prompt with "say again". The next series of program steps follows:

```
90 RECOG 3
100 B=PEEK(151)
110 IF B>18 THEN:SPEAK9:GOTO90:REM SAY AGAIN
120 IF B=18 THEN 10:REM RESET
130 IF B=16 THEN 160:REM ON COMMAND
140 SPEAK 10:PAUSE 2: SPEAK 14:REM ADVISE OFF
150 GOTO 170
160 SPEAK 10:PAUSE 2:SPEAK 13:REM ADVISE ON
170 SPEAK 11:REM SAY 'OK TO ACT'
180 RECOG 2
190 C=PEEK(151):REM SAY YES OR NO
200 IF C>9 THEN:SPEAK 9:GOTO 180:REM SAY AGAIN
210 IF C=9 THEN 10:REM RESET
.........
```

At this point we are ready to throw a switch. The value of B is 16 for on and 17 for off. The balance of the program follows:

```
220 RECOG 1:REM GET SWITCH NUMBER
230 D=PEEK(151)
240 IF D>7 THEN:SPEAK 9:GOTO 220:REM SAY AGAIN
250 E=PEEK(56577)
260 IF C=17 GOTO 300
270 POKE 56577,E OR 2↑D:REM ACTUAL TURN ON
280 SPEAK 12:PAUSE 2:SPEAK 13:REM ADVISE ON
290 GOTO 320
300 POKE 56577,E AND (255-2↑A):REM ACTUAL TURN OFF
310 SPEAK 12:PAUSE 2:SPEAK 14:REM ADVISE OFF
320 GOTO 90:REM GET ANOTHER COMMAND
330 END
```

The last part where we actually operate the switch has no escape words. We simply ran out of partitions! We could get more with BLANK and UNBLANK of one not fully used, and insert another RECOG command for a final verification. Or we could "sacrifice" one of our 8 switch commands for an escape.

When we want to de-activate the system we say "reset" at the proper time. After operating a switch, the program goes back for another command without first resetting. But there is an escape to reset when the system is waiting for a command.

We could have each command be a sequence of words, or one word repeated, in order to increase reliability as we did to initially activate the system. We might also want time limits at each stage of the process similar to that suggested for the initial activation. The variations are limitless.

What about those relays? A basic simple relay has a 12 volt DC coil requiring about 80 milliamperes and contacts (perhaps 2 pole, 2 throw) rated at 10 amperes at 250 volts AC or DC--enough for a 2 horsepower electric motor. Each of the 8 lines goes to one relay with the coil operated through one small transistor, plus a diode to reduce transients, and one resistor which avoids excessive drain on the power source in the computer and also helps to isolate unforseen voltages that might enter the computer from the switched circuits. The circuit for each of the 8 lines is shown. A separate 12 volt DC supply is suggested (or a battery).

There are other kinds of relays. One is an impulse relay that steps back and forth between two switch positions, or which steps around a set of several contacts. This kind of relay has the advantage of not requiring power when it is set; only for the switching action. The signal sent from the computer must then be formed into an on-off pulse with a duration of only a small fraction of a second. The shortcoming of this kind of relay is that there is no absolute means for reporting back to the computer which position it is in. There also is a two coil relay which latches in one position when one coil is energized (with a short pulse of current) and in the other position when the second coil is energized. The question as to which switch postion is active does not arise, but two separate digital signals are needed for each relay.

A "Rube Goldberg" Scheme: In this example, we are given a "black box" with a few keys which control something. We don't know what is inside the box. Perhaps we don't care. The approach here is to construct an artificial computer controlled "finger". We can literally push on the key with a solenoid (or even a pneumatic piston). We locate an "actuator" over each key to be pressed.

In this case, we don't simply have the output bit go on and stay on. Rather, we pulse it, like with the latching relay mentioned above. In order to do this, we first cause the output to become 1, then a short time later, using PAUSE (or other time wasting method), we simply reset the output to zero. A solenoid is nothing more than a (specialized) relay driving coil with a push rod on it instead of a switch contact.

Don't laugh at the seemingly "silly" solution. In the real world, there are many instances where opening up an existing machine with control buttons is not possible or not allowed (perhaps not even legal!). But that should not prevent you from installing numerical control!

An electrical variation to the solenoid scheme may be possible if you can get at the keyboard and find out how key closure is done. If, for example, pushing a key simply connects a point held at a positive voltage to ground, then the same circuit shown as a relay driver will operate if the relay load between the collector and + 12 volts is attached instead between key contacts.

APPENDIX K: 128 MODE PLAYBACK

The disk contains a speech playback program for use on the C128 in the "128" mode. The program is short and resides below your BASIC program. A BASIC demo program, DEMO-128, will load the machine language playback program which is called PLAY-128 and then the Voice Master vocabulary "ENGLISH". Then this entire vocabulary will be spoken back. Each individual word will be produced twice at two different clock rates. The program is self-documented and can be listed. Speech quality is better at the higher clock rate (FAST). In the 40 column display mode, but not in the 80 column mode, the screen will blank during speech playback. If desired, keep the screen on in the 40 column mode with the SLOW command, which reduces the processor clock speech to 1 Mhz.

Instructions for using PLAY-128 are given in REM statements in the DEMO-128 program. This program must be RUN"DEMO-128" when in the "128" mode. It is not necessary to load PLAY-128 from DEMO-128.

A speech file is loaded with BLOAD"filename",B1 which loads the file into bank 1 (upper 64K). Speech will always load at location $14000, so keep this in mind when creating the vocabulary in the "64" mode. Prior to recording speech (always from the "64" mode), use the CLEAR64 command (which is automatic on power up as the default page setting).

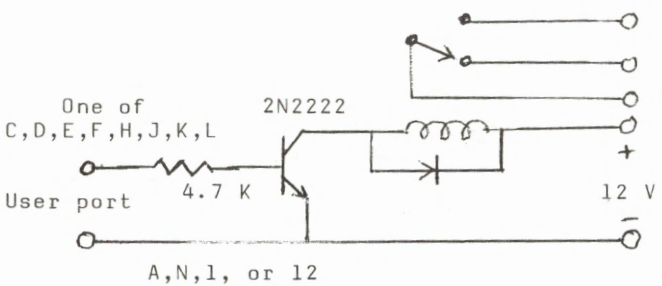
Because 128 BASIC also uses BANK 1 for variables and strings, there is the possibility that it will conflict with speech data, especially if the vocabulary is large. After loading PLAY-128, do the following from the keyboard or as a numbered BASIC statment in your program:

```
BANK1:POKE51,0:POKE52,64:BANK0
```

which sets the end-of-array pointer to $14000. After you have loaded the speech file, do the following:

```
BANK1:POKE53,PEEK(16640):POKE54,PEEK(16641):BANK0
```

which sets the bottom of string storage to the end of the speech file. This second set of commands must be done each time you load in another vocabulary because vocabulary lengths differ.



COVOX INC.
675-D Conger Street • Eugene, Oregon 97402 • U.S.A.
Area Code (503) 342-1271 • Telex 706017 (Av Alarm UD)